PIVOTAL: DEVELOPMENT GUIDE V0.94

Database and programming design

Sean Cross
Sean.cross@crm.co.nz

Contents

Document Revision

Date	Author	Comments
20 October 2014	Sean Cross	Added user permissions section.
20 August 2014	Sean Cross	Added EML dev urls
25 July 2014	Sean Cross	Minor tidy up
18 July 2014	Sean Cross	Added section on logging
22 April 2014	Sean Cross	Using enums in Lightspeed
17 April 2014	Sean Cross	Updated file and svn paths Added Raygun notes Updated urls Included programming and testing notes
25 March 2014	Hoang Duy Anh, Nguyen	Added Kendo upgrade notes
29 Jan 2015	Sean Cross	Added further UI testing notes
19 March 2015	Sean Cross	Added Git notes and development process
23/4/2015	Sean Cross	Added notes on getting the next PK Id
19/6/2015	Sean Cross	Profiling section
19/8/2015	Sean Cross	VC account URL details

Glossary

ACC	Accident Compensation Corporation. NZ Government responsible for managing Injuries and Weekly Compensation For historic reasons a large number of areas and items in the software are called, or prefixed, ACC. In current use, ACC should be interpreted as Injury Management
Entity	.net object capable of being loaded or saved to a database. E.g. User, AccClaim
IM	Injury Management
Jurisdiction	Area or region with a distinct set of regulations and requirements. E.g. New Zealand, NSW, Victoria
Member	Employee or injured worker. Anyone who can have a claim or occupational test done.
WorkCover	Generic term for Australian workers compensation scheme

Sean's rules for development

Delivery is the most important thing. Quality is also important but without delivery, it's irrelevant

YAGNI: You ain't gonna need it.

DRY: Don't repeat yourself

Code should be as simple as possible to do the job but no simpler

Unit test all the things

Use existing libraries where possible

Useful links and contacts

Pivotal Australia Test Version (updated each sprint)	https://pivotaltestau.catalystrisk.co.nz (NZ server) http://PivotalTest (EML server)
Pivotal Australia Daily Version (updated each day)	https://pivotaldailyau.catalystrisk.co.nz/ (NZ server) http://PivotalDaily (EML server)
Pivotal Australia Build Version (updated each successful build)	https://pivotalbuildau.catalystrisk.co.nz/(NZ server) http://PivotalBuild (EML server)
Pivotal New Zealand Daily Version (updated each day)	https://pivotaldailynz.catalystrisk.co.nz/
Developer sandbox (updated as required)	https://pivotalsandbox.catalystrisk.co.nz/ (NZ server) http://PivotalSandbox EML server)
Username/Password for Test/Build:	Admin/allblacks or AuTest/allblacks
mailing list : **devs/managers/testers** **devs/testers**	 PivotalDev@catalystrisk.co.nz PivotalDevelopers@catalystrisk.co.nz
RedMine – Project Tracker **Active items** **Current Sprint** **Weekly Comp**	 https://10.8.0.25/redmine/projects/woolworths/issues?set_filter=1 https://10.8.0.25/redmine/projects/woolworths/issues?query_id=226 https://10.8.0.25/redmine/projects/weekly_compensation
Build server administration	http://202.36.68.67/
Raygun error tracking	https://app.raygun.io/dashboard/6gh9ke

Catalyst Applications

There are a number of different applications that go together to form the complete claims management system.

Pivotal

Pivotal is a web based claims management application. It is written in Asp.net MVC. It has nearly complete case management functionality but limited finance and administration functionality. It includes the front end to the Document Management System.

It uses the Claims, CmsShared and Documents databases.

Welcome sean! [Settings] [Change Password] [Log Off]

Home | ACC | Occ. Health | Provider | Members | Documents | Timesheet | Wiki | Admin | Search: GO

Active claims | Recent | To-Do | Approval Requests | Claim Forms | Client Reports | New Claim

Claim #: SAMPLE - **ACC 45:** A129456
Claimant: Sean Cross
Client: Sean corp

Add Note Highlight Add Note

General | Notes | Medical | Approvals | Milestones | IRP | Invoices | Income Comp. | Estimates | QA | Documents

Claim details

ACC 45	A129456 (View)	Case Manager	Sean Cross
Status	REOPEN	Status Details	--
Received	15/07/2010 1103	Received by ACC	14/07/2010
Decision due	12/09/2010		
Case Type	C2 - High risk - Gr...	Risk	Low

Injury Details

Injury date	03/07/2009	Time	12:47 p.m.
Code 1	S34..	Side	Both
Code 2	SE01.	Side	Left
Code 3	SE01.	Side	Left
Type	Fracture	Site	Ankle
Guideline	Fracture, bimalleolar (View)	Timeframe	Min 7, Opt 14, Max 42
Diagnosis	Fracture Ankle	CM Update	This is a tribute to the greatest song in the world!
Was Fatal	☐		

Personal details

Name	Sean Cross	nee	
Date of birth	2/04/1982	Gender	Male
Address	45C Bibby Street Waipawa Central Hawke's Bay 4210	Work phone Home phone Mobile phone	06 835 5868 08 123 4567 021 123 456
Email	sample@crm.co.nz		

Employment Details

Client	Sean corp	Cost Centre	s101: Auckland
Supervisor	Smith, David View New	Manager Report	Work Injury - Acce...
Job Classification	Sedentary		

Provider

Provider	Cross, Sean (Mr): Crossworx	Seen	3/07/2009
First Incapacity	03/07/2009	Expected RTW	31/05/2012
Details	Injured in dispute!		

Save Changes | Cancel

Save Changes

Tasks
Letters
Supervisor Letters
Case Manager Reports
Request ACC History
Disputes
Managers Injury Report
ACC413 Transfer Form

Flags & Warnings
Duplicate claim
Read member notes

To Do
Claim assigned to Sean Cross.
Changed todo!

Dates

Opened	22/09/2009
Closed	
Accepted	
Cover decision	31/08/2009
Updated	10/07/2013
RTW	

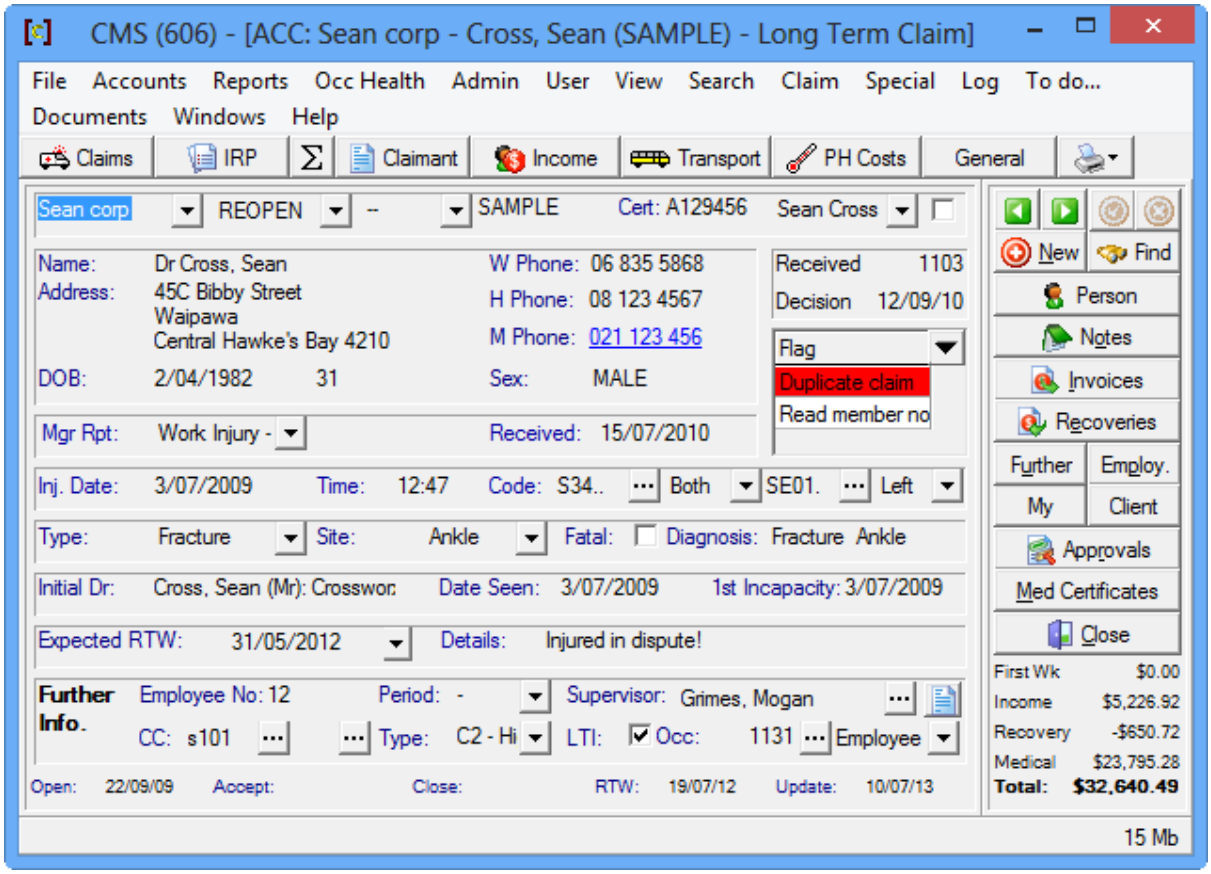
Costs

Income	$5,226.92
Recovery	-$650.72
Medical	$28,064.28
Total	$32,640.49

CMS

CMS is Catalyst's original Claims Management Software. It has been developed since 1995. It is a desktop application delivered via citrix. CMS is now being phased out and replaced by Pivotal. All case management is now handled via Pivotal but CMS is currently still used for finance and administration.

CMS uses the Claims and the CmsShared databases.



CMS/Pivotal Comparison

Feature	CMS	Pivotal
Type	Desktop	Web application
Delivery	Citrix	Web
Development framework	Delphi 2007	C# ASP.net MCV
Claims Management	X	X
Occupational health	X	X
Finance	X	
User Administration	X	X
Client Administration	X	
Letter Template and Report creation	X	
Sending automated reports	X	
Payment Processing	X	
Letter editing	X	X
Invoice Entry	Bulk and claim	Claim entry only
Document Management		X
Medical Provider login		X

Envisage

Envisage started out as a document management web application. It has now become a web service and backend for document management with the front end now done in Pivotal.

Report server

This is a windows server that creates reports for Pivotal. As the original reports are written using Delphi components, they cannot be directly created in Pivotal. Therefore Pivotal calls the Report Server to retrieve the required reports.

Docs Converter

A command line application that converts Word and image files into PDF for the document management system. It runs every 10 minutes in the background.

Emailer and CmdEmailer

As part of the reporting process, a record is placed in the EmailQueue table for each report that needs to be sent out. The Emailer and CmdEmailer programs process the queue and send the emails. CmdEmailer is a replacement for the original Emailer app.

CmdProcessor

A command line application that reads the ReportQueue table and creates reports as required. Created reports are then added to the EmailQueue table.

Technology Comparison

Application	Programming language	Application type	Status
CMS	Delphi 2007 VCL	Desktop (via citrix)	Being replaced by Pivotal
Pivotal	C# 5 (.net 4.5) ASP.net MVC 4 Silverlight (letter editing only)	Web App	Active, in development
Envisage	Delphi 2010 Intraweb	Windows Service	Active
CmdEmailer	C# 5	Console App	Active
Docs Converter	C# 5	Console App	Active
CmdProcessor	C# 5	Console App	Active
Report Server	Delphi 2007	Windows Server	Active but not being developed

Version Control

Pivotal and CMS use Subversion version control. There are 2 repositories in use:

Repository	Projects
https://secure2.svnrepository.com/s_scross/CrmDev or http://catalystrisk.svnrepository.com/svn/CrmDev	CMS, Envisage, all other Delphi projects
git@catalyst.sourcerepo.com:catalyst/PivotalGit.git	Pivotal, all other .net projects
https://secure2.svnrepository.com/s_scross/DevBin or http://catalystrisk.svnrepository.com/svn/DevBin	Binary dependencies such as CMS, Report server and Envisage for use on CI machine
https://secure2.svnrepository.com/s_scross/Database/ or http://catalystrisk.svnrepository.com/svn/Database	SQL Version control scripts for database
https://secure2.svnrepository.com/s_scross/Documents or http://catalystrisk.svnrepository.com/svn/Documents	Project documentation
https://secure2.svnrepository.com/s_scross/Pivotal/branches/release or http://catalystrisk.svnrepository.com/svn/Pivotal/branches/release	Release branch
https://secure2.svnrepository.com/s_scross/Pivotal/branches/test or http://catalystrisk.svnrepository.com/svn/Pivotal/branches/test	Test branch (for sandbox)

User accounts are setup/modified at the following addresses:

SVN https://secure2.svnrepository.com:3001/

GIT https://secure18.sourcerepo.com/

For Git private key

1. Run a command prompt as administrator
2. Run
   `cd %userprofile%/.ssh`
3. If you see "No such file or directory", please let me know
4. Run the following
   "C:\Program Files (x86)\Git\bin\ssh-keygen" -t rsa -C "your_email@example.com"
5. Enter the following responses
   a. For the filename type id_rsa
   b. Overwrite existing file
   c. No passphrase
6. Run the following to copy your public key
   `clip < id_rsa.pub`
7. Paste key in user account at https://secure18.sourcerepo.com/

Continuous Integration

Development Process

Process

Overview

Source control will move to git.

Development will work in 2 week sprints. This will start with a planning session, followed by development, testing, release testing and release.

Features and changes will be broken out into separate branches and merged back in when complete.

The git branching model is taken from http://www.diaryofaninja.com/blog/2014/09/11/so-you-want-your-team-to-start-using-git-ndash-part-4-team-workflows and http://nvie.com/posts/a-successful-git-branching-model/

Planning

At the start of each sprint, we will do planning and estimating. This will involve going through new and outstanding tasks, estimating how long they will take, seeing if additional details are required and allocating to categories. Tasks that are to be completed in the current sprint will go into the 'Current Sprint Category'. Tasks that are to be completed next, go into the 'Next Sprint' category. Otherwise tasks are categorised into other categories.

Development

When developers are assigned a task, they create a feature branch from the develop branch. All changes are checked into this branch. Once the task is complete, the feature branch is merged back into the develop branch and the task is assigned for testing.

If all planned features are completed before the end of the sprint, developers can:

- work on bug fixes, hot fixes as advised
- work on tasks from the Quick category
- review tasks in the Next category and request additional information as required

Note: developers should resolve outstanding bugs before starting new features

Testing

Testers will have as many test environments as required. See Environments section below.

At the start of each sprint, testers can review planned tasks, ask for additional details, plan test cases and the like.

As features are implemented, the development branch will be updated and this will update the build and daily environments.

Testers will test each task and either assign it back to the developer for additional work or approve it for release.

Once the release branch is updated, testers will perform final testing and regression testing against the release test environment.

Release

A release branch is taken from the development branch partway through the sprint. Once this is taken, no new features should be added to the release branch. Bug fixes for release features should be merged into the release branch.

Merging into the release branch will update the release test environment. Testers should do final testing against the release test environment.

Once the release branch has been tested and signed off, it will be merged into the master and development branches.

Live environment will be updated from the master branch. In general the live environment will be updated approximately once per sprint.

Task Management

Task Categories

- Current Sprint
- Next sprint
- Quick
- New
- Later, Release 2 etc

New tasks will be assigned to the New category. Once tasks are reviewed they will be assigned to another category as appropriate.

Tasks for the current sprint will be assigned to the Current Sprint category.

Template

A standard template will be used to make sure that all required items are included in the task description. A sample template is as follows:

```
Title:

Priority:

Category:


*Description*



*Database*



*Pivotal*



*Unit Tests*



*UAT Tests*
```

Source control

Move to git

See

- http://www.diaryofaninja.com/blog/2014/08/20/so-you-want-your-team-to-start-using-git-ndash-part-1-getting-started
- http://www.diaryofaninja.com/blog/2014/08/20/so-you-want-your-team-to-start-using-git-ndash-part-2-pushing-it-up-somewhere
- http://www.diaryofaninja.com/blog/2014/08/27/so-you-want-your-team-to-start-using-git-ndash-part-3-more-than-just-committing
- http://www.diaryofaninja.com/blog/2014/09/11/so-you-want-your-team-to-start-using-git-ndash-part-4-team-workflows
- http://nvie.com/posts/a-successful-git-branching-model/

Source Tree installed is checked into svn in C:\DevDrive\DevBin\Tools\SourceTreeSetup_1.6.12.exe.

Database changes

Database changes should be performed by DBA or Sean, and signed off by Sean

Environments

Testing and production environments will create created as required. Suggested environments are:

Environment	Server	Branch	Data	Updated
Build https://pivotalbuildau.catalystrisk.co.nz/	NZ Test	develop	Sanitised real Au	Every develop check in
DailyAu https://pivotaldailyau.catalystrisk.co.nz	NZ Test	develop	Sanitised real Au	Daily
DailyNZ https://pivotaldailynz.catalystrisk.co.nz	NZ Test	develop	Dummy NZ	Daily
DailySandboxAu https://pivotalsandbox.catalystrisk.co.nz/	NZ Test	develop	Sanitised real Au	Every Sandbox tagged check in
ReleaseTest https://pivotaltestau.catalystrisk.co.nz	NZ Test	Release	Sanitised real Au	Manual
Production https://pivotalproductionau.catalystrisk.co.nz	NZ Test	Master	Sanitised real Au	Manual
PivotalTest http://PivotalTest	Au Test	develop	Real Au	On demand
Business UAT http://PivotalUAT	Au live	Release	Real Au	On demand
Training http://PivotalTraining	Au live	Master	Real Au	On demand
Live http://Pivotal	Au live	Master	Real Au	On demand

Workflow

We use the “GitFlow” workflow.

At a high level, GitFlow is the use of Git's branching strategy to take care of a need for hotfixes, releases, development and team feature branches.

See http://www.diaryofaninja.com/blog/2014/09/11/so-you-want-your-team-to-start-using-git-ndash-part-4-team-workflows

“master” branch

Your current latest production ready codebase at any time resides in the “master” branch. This is only merged into once a release has successfully gone out and is tagged for easy rollback. The master branch is used to update the UAT and production environments

“develop” branch

Your “develop” branch is your currently integrated work in progress branch. The develop branch is used to update the test environments

Feature branches

Feature branches are used for all new work. Each Task should be in its own branch. Once the task or feature is complete, the feature is closed and merged into develop.

“release” branches

Release branches are used for release testing, pre-release tweaks/integration and update the ReleaseTest environment(s)

It’s not rare for you to work towards a release and have to make some last minute tweaks against your highly integrated “develop” codebase. For this we create a branch of “develop” and use it to pre-merge “master” into it resolving any last minute kinks while allowing your team to carry on.

“Hot fix” branches

Support to work on hotfixes as a first class citizen

Used to quickly update the production environment (“master” branch)

Tagging

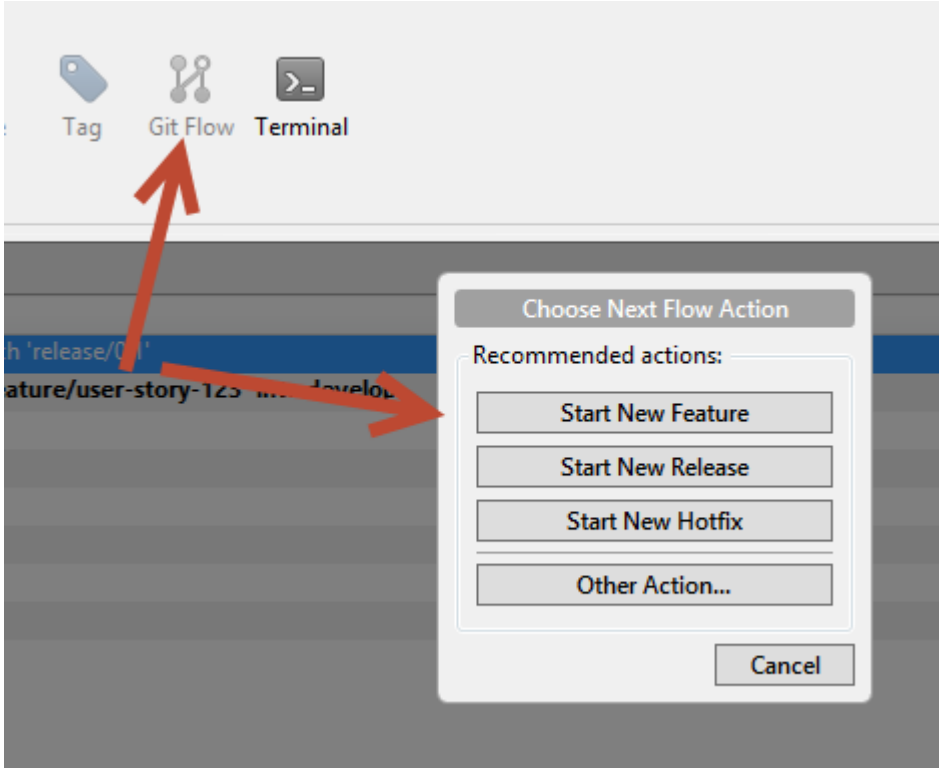
Tagging releases with semantic versioning conventions make rolling back a piece of cake.

Implementing a feature

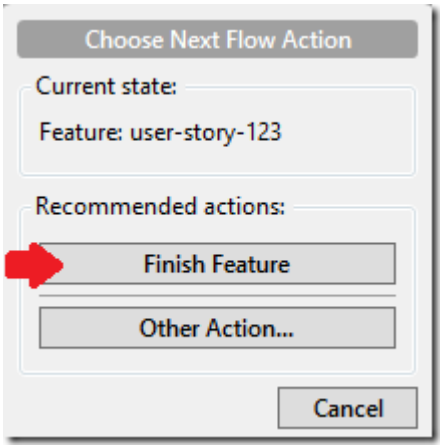
To implement a new feature or bug fix,

1. Check out develop branch (right click and click on “checkout”
2. Update code from repository (Pull changes from origin)

3. Click on the Git Flow button and choose “Start new Feature”



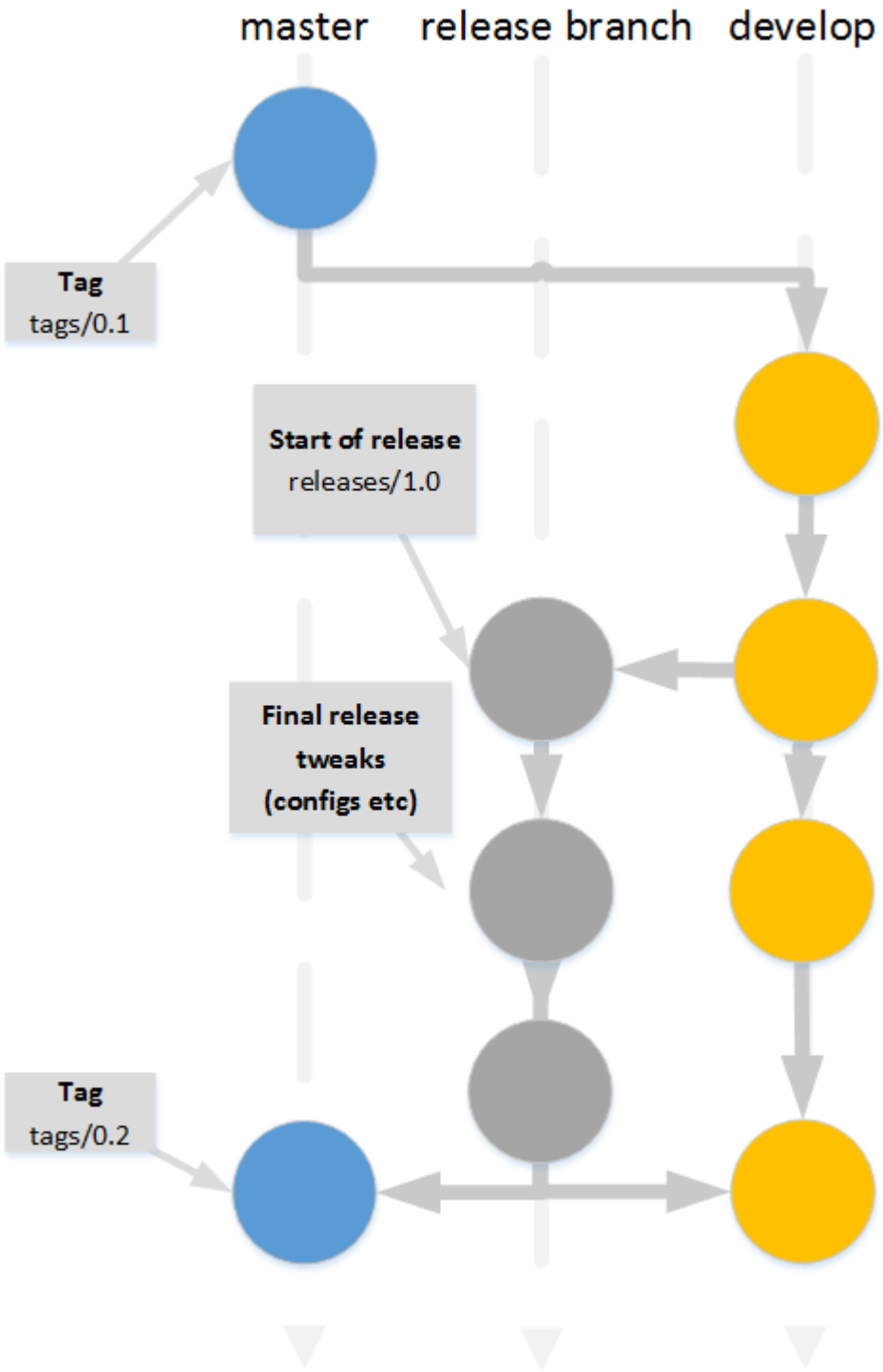
4. Give it a name such as “PIV-123”
5. Work on it…
6. Commit changes as required…
7. When changes are finished, click on GitFlow icon and click on Finish Feature
   This will update the develop branch with the changes (and the Test environments)



Creating a release

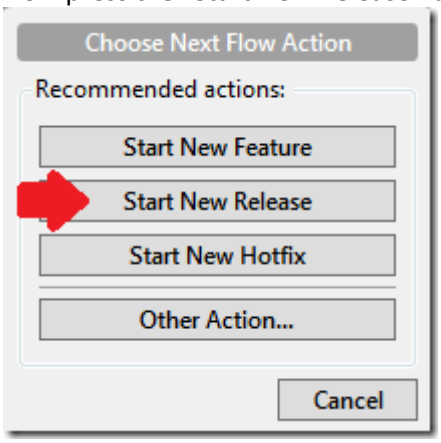
We will create a release near the end of each sprint. A release is a branch from develop and is used for regression testing etc. This allows the release to be fixed and updated without bringing in more changes from other features. Release branches update the ReleaseTest environments.

Any bug fixes for ReleaseTest issues need to be done on the current release branch.

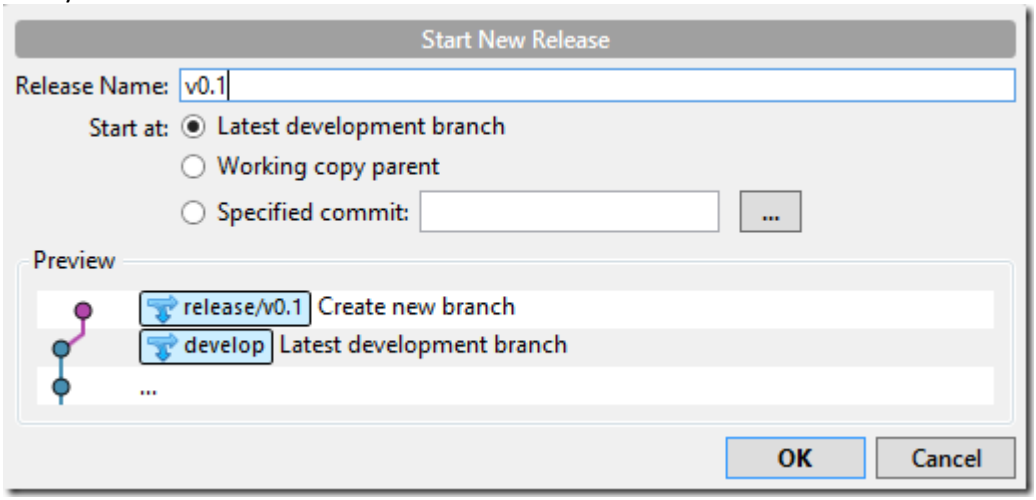


1. Checkout your “develop” branch by right clicking on “develop” and selecting “checkout “develop branch”.
2. Now click on the “GitFlow” icon.

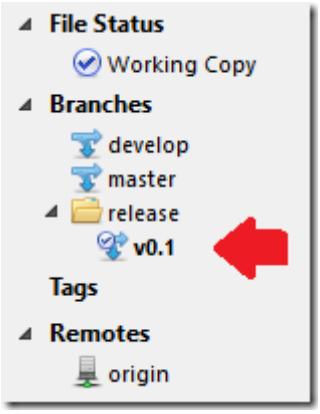
3. Now press the “Start New Release” button.



4. Give your release a name.

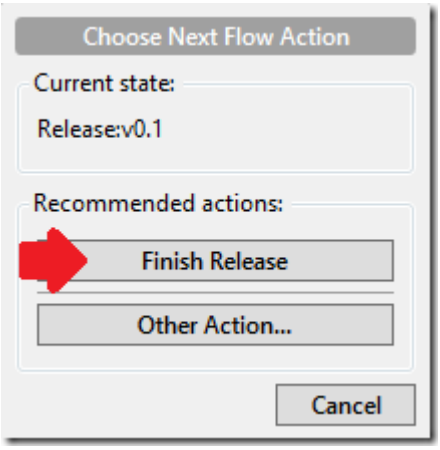


You are now checked out into your new release branch.

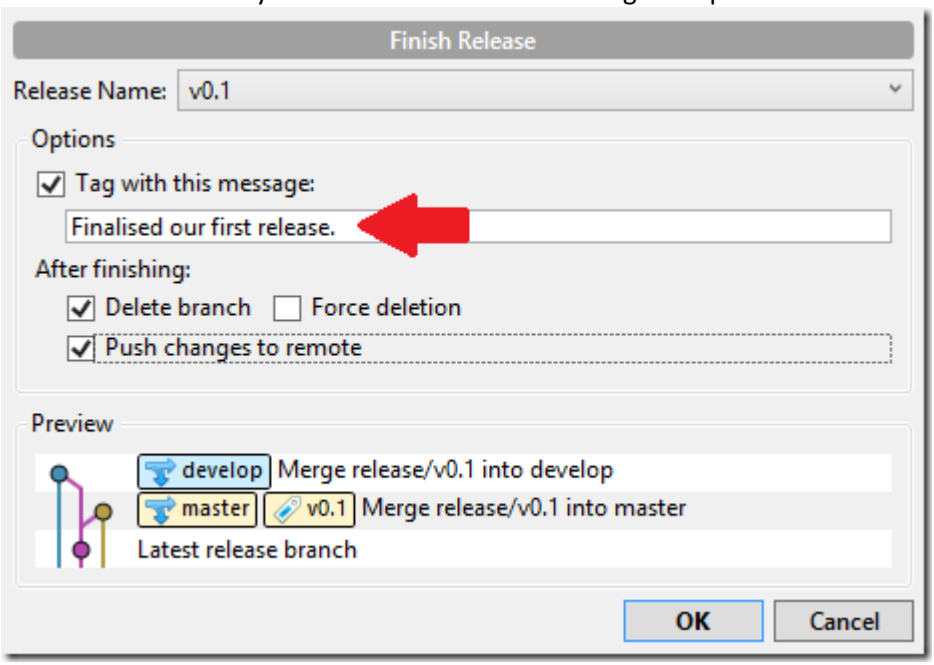


5. Push release branch to the server for regression testing
6. Make changes as required based on testing feedback.
7. When you are done with your release changes, you finish your release by again pressing the GitFlow icon.

Press the “Finish Release” button.



8. Enter a comment for your release finalisation message and press OK.



Other workflows

See http://www.diaryofaninja.com/blog/2014/09/11/so-you-want-your-team-to-start-using-git-ndash-part-4-team-workflows for other workflows such as hot fixes,

Project Management Software

Virtual Machines

The development environment for CMS and Pivotal is stored on 3 virtual machines while the source code and data is stored in a separate virtual disk drive.

SharedHdd (SharedHdd.vmx)

Contains source code and database for all CMS applications. The vmx file is loaded as the D drive in CrmDev virtual machines.

CrmDev (CmsDev.vmx)

Contains the development environment for CMS and older Delphi applications.

- Windows XP
- Delphi 2007
- Stored in C:\Virtual Machines\CrmDev\.
- Used for developing
    - CMS
    - Report server
    - Other (obsolete) apps include Emailer, Enable,

PivotalDev (W8x64S.vmx)

Contains the development environment for all .net applications, and newer Delphi applications

- Windows 8
- Delphi 2010
- Visual Studio 2013
- SQL Server Express 2008
- Used for developing
    - Pivotal (VS)
    - Envisage (D)
    - CmdEmailer (VS)
    - Docs Converter (VS)

Source code is stored in the C:\DevDrive\ folder (also mapped as D:\)

Folder hierarchy

The SharedHdd virtual disk is mapped as the D drive in the development virtual machines.

Significant folders are:

Folder	Type	Details
C:\ DevDrive\Databases	SQL Server	Database files and logs
C:\ DevDrive\dev\Components	Delphi	Catalyst and 3rd party component libraries
C:\ DevDrive\dev\GitHub	Various	Open source libraries
C:\ DevDrive\dev\Images		Images used in various projects
C:\ DevDrive\dev\Projects	Delphi	
C:\ DevDrive\dev\scripts **C:\ DevDrive\Documents\Scripts**	.sql	Scripts used for creating and modifying databases
C:\ DevDrive\trunk	.net	Base .net dev. folder
C:\ DevDrive\ trunk\Components	.net	Catalyst backend libraries and 3rd party libraries
C:\ DevDrive\trunk\Projects	.net	Catalyst applications (front end) such as Pivotal and
C:\ DevDrive\Documents	Project documentation	Open source libraries

Databases

The database server is SQL Server 2008 R2.

Catalyst and some IT clients use the same set of databases. However some IT clients using CMS/Pivotal (Assure, NZ Defence Force (NZDF) etc.) use their own copies.

There are 3 main categories of database used by Catalyst applications:

- CmsShared: Contains primarily read-only lookup data that is used by all clients such as Post Codes.
- Claims: Contains the bulk of the claims management and occupational health data. Typically called Cms**XXXX** or C**XXXX**
- Documents: Contains the document management metadata. Documents themselves are stored in the file system.

The Catalyst specific databases are called CmsDB, and Documents. IT clients with their own databases use the same schema, subject to versioning differences, but contain client specific data. E.g. NZDF has the databases CNZDF and DocsNZDF.

Table Primary keys

Table PKs are mostly integers, called xxxxId (Clients.Id, AccData.ClaimId etc).

For tables that are primarily read only in Pivotal, the PK should be an identity field. In the LightSpeed model, the tables IdentityMethod property should be set to IdentityColumn.

For tables that are read/write in Pivotal, the PK should **NOT** be an identity field. In the LightSpeed model, the tables IdentityMethod property should be set to Default.

Tables in CmsShared are almost always read only in Pivotal and should have an identity field.

Getting new Id values

It is often necessary to get new Id values; e.g. when adding new records in SQL or code without light speed.

In C#, this can be done by calling the IAccRepository.GetNextId () method.

In SQL, it is done using the sp_GetNextId_Output stored procedure.

E.g.

```
      DECLARE @BaseId int;
      DECLARE @RowCount int;
      select @RowCount = 1; -- number of Ids required, e.g. (select
COUNT(*) from #NewMembers);
      exec [sp_GetNextId_Output] @RowCount, @BaseId OUTPUT;
```

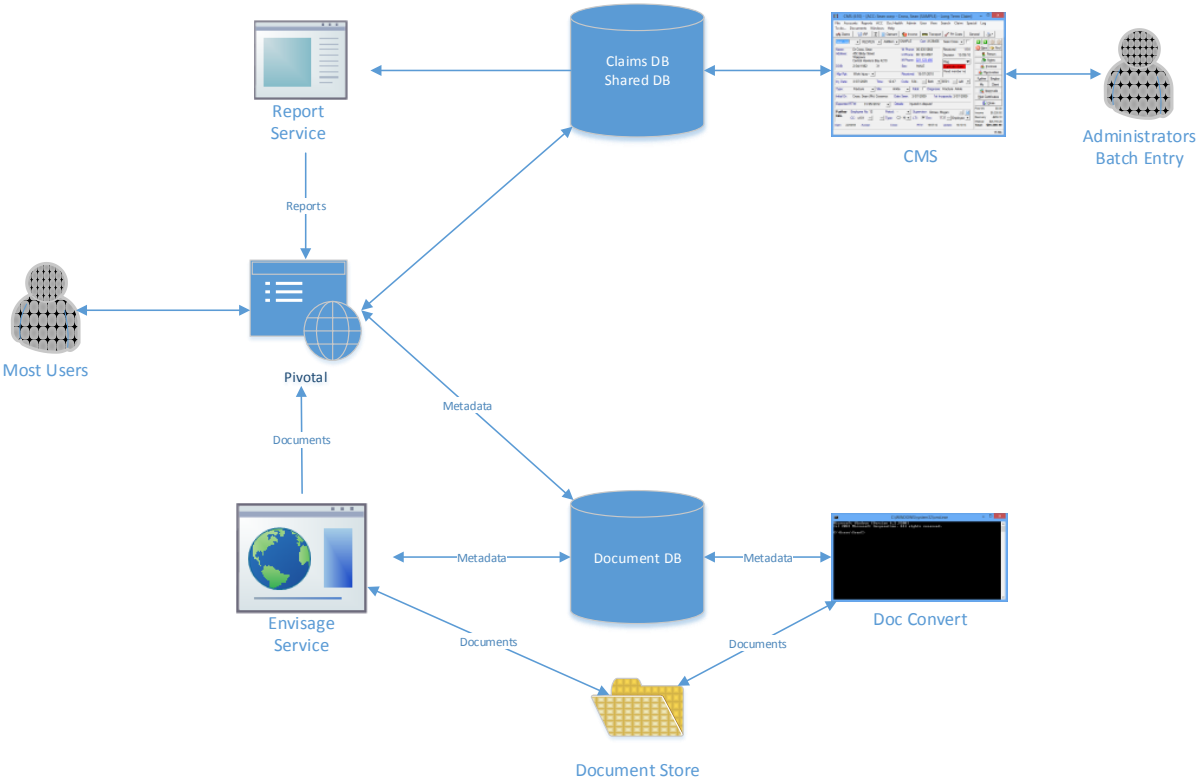
You can then use the @BaseId as your next Id. If you need to add more than one record, alter @RowCount accordingly.

Architecture

CmsWeb, the main project, is primarily a client/server web application but uses additional tiers for report created and document retrieval.

CMS is a client/server desktop application, delivered via citrix.

Envisage is a client/server web service. Document metadata is stored in the database and the documents are stored in the file system.



Pivotal

Projects

The Pivotal solution file is saved at C:\DEVDRIVE\dotnetdev\CmsWeb.sln or C:\DevDrive\dev\dotnet\CmsWeb.sln.

It contains the following projects:

Project	Type	Notes
CatalystUtils	Library	Misc. utility functions
CatalystWebUtils	Library	Misc. utility functions for web apps e.g. Html sanitiser
CmdEmailer	Console App	Console line app that sends all unsent items from EmailQueue table
CmsModels	**Library**	**Data models and business logic for Pivotal/CmsWeb**
CmsProviders	Library	ASP.net Membership and Role providers
CmsProviderTests	NUnit test library	Unit tests for CmsProviders
CmsTests		Unit tests for CmsModels, CmsWeb, CatalystUtils and CatalystWebUtils
CmsWeb	**MVC 4**	**The primary "Pivotal" project. Contains the UI for Pivotal web application.**
DocsConverter	Console App	Console app that converts non-pdf Envisage documents to pdf.
DocsModels	Library	Data models for Envisage Document Management
DocsPdf	Library	Conversion library for DocsConverter
DocsPdfTests	NUnit test library	Unit tests for DocsPdf
DocsTests	NUnit test library	Unit tests for DocsModels
LightspeedUtils	Library	Misc. utility functions for LightSpeed ORM
MedTechXXX		*Deprecated - Imports NZ ACC claims directly from MedTech's Practise Management Software*
OccTestXXX		*Deprecated - Desktop app for entering health tests offline*
PortableTests	NUnit test library	Unit tests for PortableUtils
PortableUtils	Portable Class Library	PCL implementation of utility functions
RtfEditor	Silverlight	Rich Text Editor for user editing of letters
WebTests	Web test library	

Third party controls

Telerik

- Kendo UI
- MVC extensions
- Silverlight

Mindscape – LightSpeed ORM

The Pivotal projects use the LightSpeed Object Relational Mapper (http://www.mindscapehq.com/products/lightspeed). LightSpeed model definitions are stored in .lsmodel files saved in the Entities folders of the relevant project (CmsModels and DocsModels). The class source code is stored in the .cs file with the same name.

Note: Classes that can be saved/loaded from a database (or that inherit from such) are known as Entities.

E.g. The Shared model contains entities that are used in multiple areas of Pivotal such as Client and Member. The model definition is stored in CmsModels\Entities\Shared.lsModel and the source code for the classes is stored in CmsModels\Entities\Shared.cs.

Partial classes can be used to extend the database entities. These are saved in the same \Entities\ folder as the model and named ClassNameXX.cs.

The primary models are:

Model	Location/Namespace	Details
Api	CmsModels.Entities	Deprecated - Data synchronisation entities for OccTester
Cms	CmsModels.Entities	Primarily entities related to Injury Management
DocsEntities	DocsModels.Entities	Entities derived from the Documents database
Dynamics	CmsModels.Entities	Entities that are defined at runtime
Elodge	CmsModels.Entities	Deprecated - Entities
Misc	CmsModels.Entities	
OccHealth	CmsModels.Entities	Entities used in the Occupation Health area. No longer under development
Reports	CmsModels.Entities	Entitles used in reporting
Shared	CmsModels.Entities	Entities not tied to a single area
WikiEntities	CmsModels.Entities	Entitles used in wikis

Logging

Mindscape – Raygun error logging

Uncaught exceptions are logged and sent to Raygun.io. They can be viewed at

https://app.raygun.io/dashboard/6gh9ke

To add exception to Raygun manually, you can do the following:

```
Try
{
    //your code
}
Catch(Exception ex)
{
        //add exception to Raygun
        var rc  =    new Mindscape.Raygun4Net.RaygunClient();
        rc.Send(ex);
```

}

System logging

System logs are to log system initiated actions such as temp file deletion, report processing etc that cannot assigned to a user (UserLogs) or Claim (ProcessLogs). System logs are displayed in the Admin area.

Usage:

```
int logId = repository.AddSystemLog(SystemLog.Log_DeleteTempFilesName, SystemLog.L
og_DeleteTempFiles, DateTime.Now.AddHours(1));
try
{
    ...Do stuff here
    repository.CompleteSystemLog(logId, count.ToString() + " files deleted");
}
catch(Exception ex)
{
    new Mindscape.Raygun4Net.RaygunClient().Send(ex);
    repository.UpdateSystemLog(logId, DateTime.Now, ex.Message, true);
}
```

User logging

User logs are used to log user initiated actions that are not associated with a claim such as log in /out, alter/authorise bank accounts etc.

Usage:

```
repository.AddUserLog(CmsUser.UserName, [Action Name], [Action description]);
```

Process logs

Process logs are used to record claim related activity such as changes in claim status

Usage:

```
_Claim.AddLog(CmsUser, logType, message, moreDetails);
```

MVC

Routing

In general, routes are /Controller/Action/Id.

The parameter name “Id” should be used whenever referencing an entity as opposed to e.g. ClaimId, MemberId to allow for loading of required objects.

Controller hierarchy

Controllers descend from PivotalBaseController. Typically a section will have 2 controllers, a view control responsible for displaying views and a data controller responsible for returning data as JSON. Both the view and the data controller will descend from a common base controller that in turn

descends from PivotalBaseController. E.g. AccController and AccDataController both descend from ControllerAccBase.

Coding conventions

Dependency Injection

CmsWeb uses Inversion of Control and Dependency Injection to create some objects at runtime, esp. Controllers and Repositories

These can be instantiated as required using Resolve:

```
IUserRepository repository = WindsorControllerFactory.InstanceContainer.Resolve<IUserRepository>();
```

Or by adding as parameters to the constructor:

```
public ControllerBaseAcc(IAccRepository accRepository, IUserRepository userRepository)
{
    _AccRepository = accRepository;
    _UserRepository = userRepository;
}
```

Required objects

A common convention in Pivotal is to pass in the Id of a database entity in the Action parameters and then use a RequiresXXX attribute. Before the Action is called, the controller will attempt to load the entity and populate the appropriate properties.

E.g.

```
[RequiresClaim]
public ActionResult AddClaimLog(int Id, string logNotes, bool highlight, int? minutes)
{
    logNotes = logNotes.SanitizeHtml();
    bool chargeToClient = (minutes ?? 0) > 0;
    _Claim.AddLog(CmsUser, ClaimLog.LOG_USER_NOTES, logNotes, highlight, chargeToClient);
    if (chargeToClient)
    {
        _AccRepository.AddTimesheetChargeItem(CmsUser.Id, DateTime.Today, _Claim.ClientId,
_Claim.Claim, minutes.Value);
    }
    _AccRepository.SaveChanges();
    return Redirect(Request.UrlReferrer.ToString());
}
```

Attribute	Entity	Controller(s)	Properties	Notes
RequiresClaim	AccClaim	AccXXX	_Claim, _ClaimVM	
RequiresEntity	Entity	All	_Entity	Casting required to use _Entity
RequiresMember	Entity Member, MemberVM	All Member	_Entity _Member, _MemberVM	Inherits from RequiresEntity
RequiresOrder	OccHealthOrder	OccHealthXXX	_Order, _OrderVM	
RequiresWiki	Entity Wiki, WikiVM	All WikiXXX	_Entity _Wiki, _WikiVM	Inherits from RequiresEntity

Patterns

View Models

ViewModels are frequently used when providing data to the Views for display and are always used for objects that are updated by the user and passed back to the controller.

ViewModels are typically populated from the base object using AutoMapper. This allows simplification of the model and flattening of the hierarchy. Attributes are used to define validation logic (e.g. `[Required]`) and display logic (e.g. `[DisplayFormat(DataFormatString = "{0:C}")]`.

View models are saved in the ViewModels folder. They are after their base object suffixed with VM. E.g. the Medical Certificate view model shows the MedCert object so is called MedCertVM.

View models that require auto-mapping should implement the interface IViewModel and the method Setup. Setup is responsible for creating the initial map.

e.g.

```
public class MedCertVM : IViewModel
{

    public int Id { get; set; }
    ...

    public void Setup()
    {
        Mapper.CreateMap<MedCert, MedCertVM>();
    }
}
```

View models that alter entities should implement an Update method that takes the destination entity as a parameter.

Repositories

All database access is done through repositories, specifically through an implementation of a descendant of IBaseRepository such as IAccRepository. Repository interfaces are saved in the XxxModels/Abstract folders.

Typically there will be 2 implementations of a repository, a database-based one used in the application and an in-memory one used in unit testing. E.g. IAccRepository is implemented by AccRepository and FakeAccRepository. Implementations are saved in the XxxModels/Repositories folders.

In CmsWeb, repositories are instantiated via dependency injection in the constructor. E.g.

```
public AccController(IAccRepository accRepository, IUserRepository userRepository) :
base(accRepository, userRepository)
{
}
```

Repositories can be extended via helper objects to provide functionality shared between implementations. These are also saved in the XxxModels/Abstract folders. E.g. IAccRepositoryHelper.

IBaseRepository

IBaseRepository contains the base functionality shared by more specific repositories such as IUserRepository and IAccRepository.

```
public interface IBaseRepository : IDisposable, IRepository
{
    void Attach(Entity entity);
    void Add(Entity entity);
    void Remove(Entity entity);
    TEntity Get<TEntity>(object id) where TEntity : Entity;
    Entity Get(Type entityType, object id);
    IQueryable<TEntity> GetAll<TEntity>() where TEntity : Entity;
    IQueryable<TEntity> Where<TEntity>(Expression<Func<TEntity, bool>> expression) where TEntity :
Entity;
    TEntity GetOne<TEntity>(Expression<Func<TEntity, bool>> expression) where TEntity : Entity;
    IList<TypeCodeDescription> GetTcds(string typeName);

    void AddDocument(int documentId, string reference, string title, string keywords, string comments,
string user, int sourceId = 0, int categoryId = ModelConsts.Documents_CategoryId_Default);
    IDbTransaction BeginTransaction();
}

public interface IRepository
{
    // Summary:
    //     Flushes pending changes to the underlying database.
    void SaveChanges();
}
```

IUserRepository

IUserRepository is used for loading and deleting users.

```
public interface IUserRepository : IBaseRepository
{
    User GetUser(int id);
    User GetUser(string userName);
    RemoteUser GetRemoteUser(string userName);
    void DeleteUser(User user);
    void AddUserLog(string userName, string action, string details);
    IQueryable<User> GetUsers();
    IUser GetUserEntity(string userName);
}
```

IAccRepository

IAccRepository is primarily used for dealing with ACC (Injury Management) entities.

```
public interface IAccRepository : IBaseRepository
{
    AccClaim GetClaim(int id);
    AccClaim GetClaim(string claimNo);

    void AddClaim(AccClaim claim);

    IQueryable<AccClaim> _GetClaims();
    IQueryable<AccClaim> GetUserClaims(User user);
    IQueryable<AccClaim> GetUserClaims(string userName);

    IQueryable<AccClaim> GetActiveClaims(User user);

    IQueryable<AccClaim> SearchClaims(string userName, string searchTerms);
    IQueryable<AccClaim> SearchClaims(User user, string searchTerms);
```

```
    IQueryable<IrpGoal> GetClaimIrpGoals(string claim);
    IrpPlan GetIrpPlan(int planId);
    void AddIrpGoal(AccApproval approval);
    ToDoItem GetToDoItem(int id);

    IQueryable<InvoiceSearchResult> SearchInvoices(int userId, DateTime since, string searchTerms,
bool unpaidOnly);

    void UpdateMemberAddress(Member member);
    void UpdateMemberDetails(Member member);
    int CopyEstimate(int userId, int estimateId);
    AccClaimCostSummary GetCostSummary(int claimId);

    int GetUniqueId(string name, int? startingValue = null);
    int GetNextId(int blockSize = 1);
}
```

Example of usage

From public static class IBaseRepositoryHelper

```
public static IQueryable<ClaimLog> GetLogs(this IBaseRepository repository, string reference)
{
    return from l in repository.GetAll<ClaimLog>()
            where (l.Reference == reference)
            orderby l.ProcessTime descending
            select l;
}
```

From AccRepositoryIntegration integration test

```
[Test]
public void GetLogs()
{
    Assert.AreEqual(0, _AccRepository.GetLogs("sample").Count());
    DateTime processTime = DateTime.Now;

    ClaimLog log = new ClaimLog
    {
        LogType = ClaimLog.LOG_USER_NOTES,
        LogNotes = "to be or not to be",
        Style = "Bold",
        UserId = 5,
        Reference = "sample",
        IsDeleted = false,
        ProcessTime = processTime
    };

    _AccRepository.Add(log);
    _AccRepository.SaveChanges();

    Assert.AreEqual(1, _AccRepository.GetLogs("sample").Count());
    ClaimLog retrieved = _AccRepository.GetLogs("sample").First<ClaimLog>();
    Assert.AreEqual(retrieved.LogType, ClaimLog.LOG_USER_NOTES);
    Assert.AreEqual(retrieved.LogNotes, "to be or not to be");
    Assert.AreEqual(retrieved.Style, "Bold");
    Assert.AreEqual(retrieved.UserId, 5);
    Assert.AreEqual(retrieved.Reference, "sample");
    Assert.AreEqual(retrieved.IsDeleted, false);
    Assert.AreEqual(retrieved.ProcessTime, processTime);
}
```

Jurisdiction Specific business logic

Jurisdiction specific code is called using the Jurisdiction Factory. This returns an instance of IJurisdiction.

```
var result = JurisdictionFactory.Build(CmsModels.ModelConsts.JurisdictionCode.NSW);
```

```
var result = JurisdictionFactory.Build(claim.JudisdictionCode);
```

To add additional functionality, add a method signature to IJurisdiction, and then implements in the concrete objects under CmsModels\Jurisdictions\.

Name	Type	Attributes	Methods	Purpose
IJurisdiction	interface	Readonly string CountryCode; Readonly string JurisdictionCode; Readonly string Description	n/a	General type for all Jurisdiction types
JurisdictionBase	abstract Class	Inherit from IJurisdiction (Protected Set)	Currently there are no methods inside, but will be implemented in future if it is essential	handling all general business relevant to all Jurisdictions
JurisdictionAU	abstract class	Inherits from JurisdictionBase	Currently there are no methods inside, but will be implemented in future if it is essential	Australia specific business logic
JurisdictionNZ	class	Inherits from JurisdictionBase	Currently there are no methods inside, but will be implemented in future if it is essential	New Zealand specific business logic
JurisdictionNSW **JurisdictionNT** **JurisdictionQLD** **JurisdictionSA** **JurisdictionTAS** **JurisdictionVIC** **JurisdictionWA**	class	Inherits from JurisdictionAU	**Build() :** Create instance of itself (include all information of this Jurisdiction)	State specific business logic
JurisdictionFactory	class	n/a	**Build(CmsModels.ModelConsts.Jurisdicti**	Initialize all

			onCode) : Create instance of Jurisdiction base on JurisdictionCode Return IJurisdiction	Jurisdictions

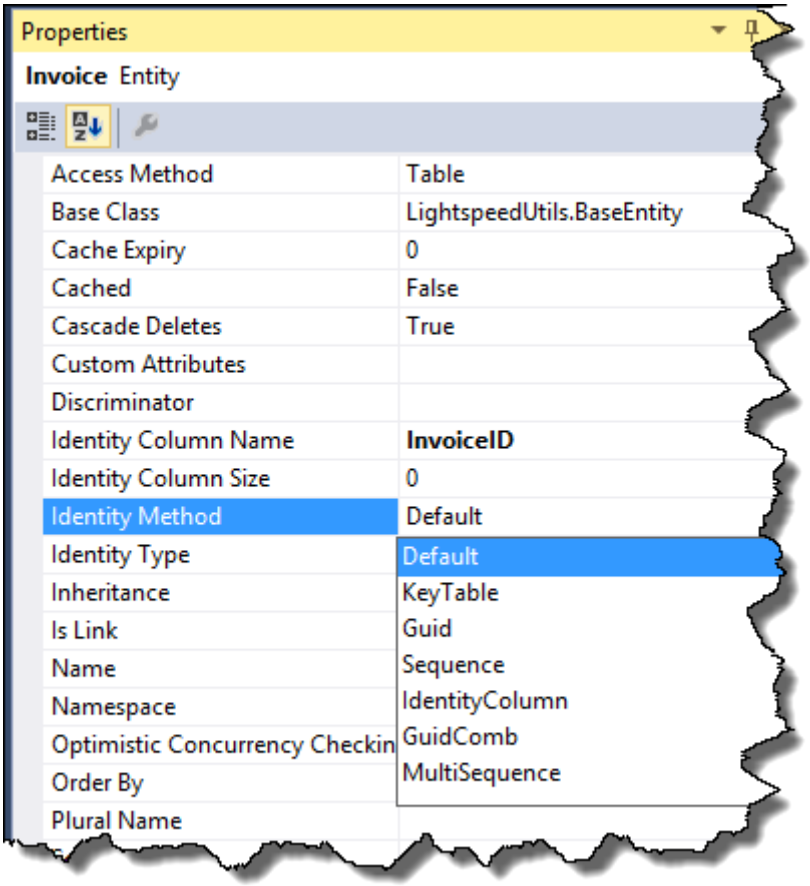
Table Id fields

Mindscape uses the primary key (Id) to load objects from the db. When creating a new object, it needs to know how to create the Id value. This is set in the Identity Method property of the object:

- If Mindscape needs to generate an integer id, Identity Method should be set to default.
- If Mindscape needs to generate an guid id, Identity Method should be set to Guid or GuidComb.
- If the database generates the Id (e.g. an Identity field), Identity Method should be set to IdentityColumn.

When creating read-only tables such as lookup tables, the PK should be an Identity field and Identity Method should be set to IdentityColumn.

When creating read/write tables, the PK should just be an int not null field and Identity Method should be set to default. Mindscape will then set the value as part of the update.



Lookup lists and Combo boxes

Boolean, Enum

Bools and Enums can be converted to a lookup list using the ToSelectList helper method defined in CmsWeb.Helpers.

E.g.

```
<% Html.Kendo().DropDownListFor(m => m.IsMale)
                .BindTo(Model.IsMale.ToSelectList("Male", "Female"))
                .HtmlAttributes(new { style = "width:75px",valign="bottom" })
                .Render();

%>

<% Html.Kendo().DropDownListFor(m => m.Risk)
        .BindTo(Model.Risk.ToSelectList())
        .UseSelectListSettings()
        .Enable(_CanEdit)
        .Render();
%>
```

Short list

For short, fixed, lookups use the help method QuickSelectList.

E.g.

```
<%: Html.Kendo().DropDownListFor(m => m.RegulatorPSide)
          .BindTo(HtmlUtilities.QuickSelectList(Model.RegulatorPSide, "", "-
", "B", "Both", "L", "Left", "R", "Right"))
          .Enable(_CanEdit)
     .HtmlAttributes(HtmlAttributes.EditorNarrowCombo)
%>
```

Dedicated tables

Dedicated tables are being phased out and replaced with TypeCodeDescription or Jurisdictional lookups. The existing tables are mostly prefixed with lu (Look Up) or Reg_ (Regulator). Other tables are:

- InjCode: Injury types
- InjrySite: Injury locations

Type Code Description

The TypeCodeDescription tables are used for generic lookups that don't require an individual table. New lookups should be created as TCDs unless other capabilities are required (e.g. jurisdictional filtering).

The primary fields are:

Field	Description
Type	The Lookup Type e.g. Values stored in TcdType.cs `CategoryType = "NTCAT";`

Code	Value stored in base table. Note a value of '>' is never returned and is used to describe the lookup data.
Description	Value shown to the user

E.g.

Type	Code	Description
APlan	>	Reasons Action plan hasn't been completed
APlan	CNotR	Claimant not returned contact request
APlan	COnly	Cover decision only
APlan	EmpC	Employer already completed
APlan	Other	Other
APlan	ROnly	Review only
APlan	TOnly	Treatment only
CaseT	>	Case Types (Favourite = Complex)
CaseT	A1	Non complex
CaseT	A2	Complex
CaseT	B1	Non complex
CaseT	B2	Complex

Items can be entered into the Claims database TypeCodeDescription or the CmsShared.dbo.SharedTypeCodeDescription tables. Types stored in the Claims database are used in preference to those stored in CmsShared.dbo.SharedTypeCodeDescription.

To retrieve the lookup values use one of

```
Xxx = repository.GetSelectList(tcdType, selectedValue);  // returns IEnumerable<SelectListitem>


Or xxx = repository.GetTcds(tcdType);  // returns IList<TypeCodeDescription>

Or

<% Html.Kendo().ComboBox()
        .Name("CategoryCode")
        .EmptyMessage("Select category...")
        .SelectedIndex(-1)
        .DataSource(source => source.Read(read => read.Action("GetTcds", "SharedData",
                new { tcdType = TcdType.CategoryType, isFilterMode = false }) ))
        .HtmlAttributes(HtmlAttributes.EditorMedium)
        .UseSelectListSettings()
        .FilterContains()
        .HighlightFirstMatch(true)
        .AutoFill(true)
        .Render();
%>
```

To get the description, use GetTcdDescription.

```
Var description = Repository.GetTcdDescription(TcdType.LitigationType,
this.LitigationType);
```

Jurisdictional

There are 2 types of Jurisdictional lookups; mapped lookups and jurisdictional specific lists.

Jurisdictional Mapped Lists

Mapped lists are used when the same lookup list is used in every jurisdiction but the codes provided to the jurisdictional authorities changes. E.g. Liability Status and Reasonable Excuse. The same lookup lists and code values are using in each jurisdiction. However when reporting, translation is provided to convert the saved code value to that required.

These are stored in the CmsSharedd.dbo.MappedLuXXX tables.

Lookup types are stored in the MappedLuTypes table. E.g.

Type	Type	Description
2	LiabilityStatus	Liability Status
4	ReasonableExcuse	Reasonable Excuse

Codes are stored in MappedLuCodes.

Id	TypeId	Code	Description
11	4	1	Insufficient medical information
12	4	2	Worker unlikely to be a worker
13	4	3	Unable to contact worker
14	4	4	Worker refuses access to information (privacy)
15	4	5	Injury is not work related
16	4	6	Injury not significant
17	4	7	Notice of injury more than 2 months after date of injury

The jurisdictional mappings are stored in MappedLuMappings. These are only used in reporting.

Id	TypeId	Code	JurisdictionCode	MappedCode	AppliesFrom	AppliesUntil
11	4	1	NSW	1	1/01/1950	1/01/2100
12	4	2	NSW	2	1/01/1950	1/01/2100
13	4	3	NSW	3	1/01/1950	1/01/2100
14	4	4	NSW	4	1/01/1950	1/01/2100
15	4	5	NSW	5	1/01/1950	1/01/2100
16	4	6	NSW	6	1/01/1950	1/01/2100
17	4	7	NSW	7	1/01/1950	1/01/2100

Adding Mapped Lookups

The easiest way to add mapped lookups is to add an entry to MappedLuPreMappings and then run sp_InitialiseMappedType. This will update existing mappings and add new mappings.

TypeName	Code	Description	NSW	NT	NZ	QLD	SA	TAS	VIC	WA	AppliesFrom
ReasonableExcuse	1	Insufficient medical information	1	NULL	NULL	NULL	NULL	NULL	NULL	NULL	1/01/1950
ReasonableExcuse	2	Worker unlikely to	2	NULL	NULL	NULL	NULL	NULL	NULL	NULL	1/01/1950

		be a worker									
ReasonableExcuse	3	Unable to contact worker	3	NULL	NULL	NULL	NULL	NULL	NULL	NULL	1/01/1950
ReasonableExcuse	4	Worker refuses access to information (privacy)	4	NULL	NULL	NULL	NULL	NULL	NULL	NULL	1/01/1950
ReasonableExcuse	5	Injury is not work related	5	NULL	NULL	NULL	NULL	NULL	NULL	NULL	1/01/1950
ReasonableExcuse	6	Injury not significant	6	NULL	NULL	NULL	NULL	NULL	NULL	NULL	1/01/1950
ReasonableExcuse	7	Notice of injury more than 2 months after date of injury	7	NULL	NULL	NULL	NULL	NULL	NULL	NULL	1/01/1950

Jurisdictional Specific Lists

Jurisdictional specific lists are used for lookups that vary over time or by claimant jurisdiction

E.g. Injury Location. In NZ this uses the NZ Regulator lookups while in Victoria it uses VCode and in most of the rest of Australia it uses TOOCS codes depending on when the claim was created.

These are stored in the CmsSharedd.dbo.JurisdictionLuXXX tables.

Each individual lookup type is listed in the JurisdictionLuTypes table. E.g.

Type	Type	Description
1	InjLocation	TOOCS Location v1.0
2	InjLocation	TOOCS Location v2.1
3	InjLocation	TOOCS Location v3.1
5	InjLocation	NZ Regulator
6	InjLocation	Not specified

Code is the name that will be used by Pivotal to locate the lookups. Description is used to differentiate the different lists that can be used for that lookup category.

Usage is stored in the JurisdictionLuUsage table. E.g.

Id	JurisdictionCode	AppliesFrom	TypeId
2	NSW	1/07/1991	1
3	NSW	1/07/2002	2
4	NSW	1/07/2011	3
6	QLD	1/07/2006	3

5	NZ	1/01/1990	5

So, NSW uses TOOCS v1.0 from 1/7/91, v2.1 from 1/7/2002 and v3.1 from 1/7/2011.

The lookups themselves are stored in JurisdictionLuCodes. E.g.

Id	TypeId	Code	Description
85	1	110	Cranium
…			
28	1	900	Artificial aids
514	2	110	Cranium
…			

Adding Jurisdictional lookups

Simple (no variations)

1. Add an entry in JurisdictionLuTypes with the code and the description. The description should indicate the restrictions on usage or where the data is derived from. E.g. 'NZ only', 'TOOCS v3.1', 'NSW' etc.
2. Execute the stored procedure sp_AddJurisdictionTypeUsage using the TypeId from JurisdictionLuTypes
3. Add entries to JurisdictionLuCodes using the TypeId from JurisdictionLuTypes

Complicated (variations for different time periods or jurisdictions)

1. Add an entry in JurisdictionLuTypes for each variation. The description should indicate the restrictions on usage or where the data is derived from. E.g. 'NZ only', 'TOOCS v3.1', 'NSW' etc.
2. Add entries in JurisdictionLuUsage for each variation showing the jurisdiction and start date date
3. Add entries to JurisdictionLuCodes using the TypeId for each variation

Checking Allocations

The allocation of lookups to jurisdictions can be checked by refreshing C:\DevDrive\DotNetDev\Documentation\PivotalAu\Jurisdictions.xlsx. The highlighted cells indicate lookups that have not been specified for the jurisdiction.

Row Labels	NSW	NT	NZ	QLD	SA	TAS	VIC	WA
Agency	1/07/2011	1/01/1900	1/01/1990	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900
DutyStatus	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900
Ethnicity	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900
InjLocation	1/07/2011	1/01/1900	1/01/1990	1/07/2006	1/01/1900	1/01/1900	1/01/1900	1/01/1900
InjuryNature	1/07/2011	1/01/1900	1/01/1990	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900

InjuryResult	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900
Mechanism	1/07/2011	1/01/1900	1/01/1990	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900
Occupation	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900	1/01/1900

Using jurisdictional lookups

Both mapped lists and jurisdictional specific lists use the same access methods.

To retrieve a lookup, use the GetJurisdictionLookupXXX methods defined on IBaseRepository. These are defined in IBaseRepositoryJurisdictionHelper.cs.

E.g.

```
public static string GetJurisdictionLookupDescription(this IBaseRepository repository, string
typeName, string jurisdictionCode, DateTime created, string lookupCode)

public static IList<JurisdictionLuCode> GetJurisdictionLookup(this IBaseRepository repository, string
typeName, string jurisdictionCode, DateTime created)

public static IList<JurisdictionLuCode> GetJurisdictionLookup(this IBaseRepository repository, int
typeId)

public static int GetJurisdictionLookupId(this IBaseRepository repository, string typeName, string
jurisdictionCode, DateTime created)
```

Lookup codes are defined in JurisdictionLookupType.cs. E.g.

```
    public static class JurisdictionLookupType
    {
        public const string InjuryLocation = "InjLocation";
        public const string InjuryNature = "InjuryNature";
         …

    }
```

To retrieve SelectItems for use in a combo box, use

```
public static IEnumerable<SelectListItem> GetJurisdictionLookupSelectList(this IBaseRepository
repository, string typeName, string jurisdictionCode, DateTime created)
```

defined in IRepositoryHelper.cs.

E.g.

```
InjuryCodeList = repository.GetJurisdictionLookupSelectList(JurisdictionLookupType.InjuryNature,
JurisdictionCode, DateOpened);

InjurySiteList = repository.GetJurisdictionLookupSelectList(JurisdictionLookupType.InjuryLocation,
JurisdictionCode, DateOpened);
```

Alternately, use `SharedDataController.GetJurisdictionLookups(string typeName, string jurisdictionCode, DateTime created)`

E.g.

```
<% Html.Kendo().ComboBoxFor(model => model.RegulatorLocation3A)
          .AutoFill(true)
          .DataSource(source => source.Read(
                    reader => reader.Action(
                              "GetJurisdictionLookups", "SharedData",

          new { typeName = JurisdictionLookupType.LocationNZ3a, jurisdictionCode = Model.JurisdictionC
ode, created = Model.DateOpened })))
```

```
            .UseSelectListSettings()
            .Filter(FilterType.Contains)
            .RequireSelection()
            .HighlightFirstMatch(true)
            .Enable(_CanEdit)
            .Render();
%>
```

User Permissions

The user permissions section in Pivotal incorporates both application permission (can I do this) and corporate hierarchy (who are my peers and who do I report to).

Tables

Table	Use	Type
Permissn	Stores users and user groups (IsUserGroup = true for user groups, false for users)	Application and Corporate
UserPermission	Allocates permissions to users. See below for Permission Types	Application
UserGroups	Allocates users to user groups	Application
UserRoles	Lookup list of roles and authorisation limits	Application and Corporate
Teams	Lookup list of teams that users can belong to	Corporate
UserTeams	Allocates users to teams	Corporate
UserJurisdictions	Determines which jurisdictions a user can view	Application

A user can belong to 0 or more user groups, and 0 or more teams.

In code

Permissions are defined in UserSecurity.cs. A user has a given permission if:

- They have been assigned that permission in User Setup
- Or they belong to a group that has been assigned that permission

In code, this can be checked using User.InRole(). E.g.

```
var canAddUser = User.IsInRole(UserSecurity.RoleAddUsers);
```

Role names are listed in the following table.

Adding new permissions:

To add a new permission it needs to be added to the following areas:

Add to the permission names consts (UserSecurity.cs):

```
public const string RoleCanEditDeleteTask = "RoleCanEditDeleteTask"; //75
```

Add to the permission ids consts (UserSecurity.cs):

```
public const int RoleCanEditDeleteTask_id = 75;
```

Add to AllPermissions array (UserSecurity.cs):

```
public static PermissionDetail[] AllPermissions =
{
...
    new PermissionDetail() {PermissionId = RoleCanEditDeleteTask_id, PermissionName = RoleCanEditDeleteTask,
            GroupName = ACC_Group, Description = "Can Edit/Delete Task", Visible = true},

};
```

Check for permissions as required:

```
if (_CanEdit && Page.User.IsInRole(UserSecurity.RoleCanEditDeleteTask))
{
    ...
}
```

Permission Types

Group	Name	Description	Notes
Administration			The administration group has permissions relating to the setup and maintenance of Pivotal and users
Administration	RoleAddUsers	Add New Users	Can add new users
Administration	RoleAdmin	Admin	Show Admin menu and provide access to application administrative tasks that don't have separate permissions.
Administration	RoleCanAuthoriseBankAccounts	Can authorise bank accounts	Move to Financial?
Administration	RoleCanAuthoriseChanges	Can authorise reports	Not used. Hide
Administration	RoleCanEditPasswords	Can Edit Passwords	Can edit/reset user passwords
Administration	RoleCanManageStandardMessages	Can Manage Standard Messages	
Administration	RoleEditMilestoneDefinitions	Can Edit Milestone Definitions	
Administration	RoleEditUserDetails	Edit User Details	Can edit user details (name, email etc) for existing users
Administration	RoleEditUserGroupMemberShip	Change User's group	Add add/remove users from groups
Administration	RoleEditUserGroups	Add/Edit User Group details and permissions	Can edit user groups and add new ones
Administration	RoleEditUserPermissions	Edit User Permissions	Can edit permissions for existing users
Administration	RoleViewNonSelectedJurisdiction	View Non Selected Jurisdiction	User can view (read-only) claims belonging to a jurisdiction to which they are not assigned
Claims			The administration group has permissions relating to the management of claims
Claims	RoleACC	View Claims	Can View claims. Without this permission the user cannot view any claims
Claims	RoleAccQA	Quality Assurance	User can perform quality assurance tasks on a claim (QA tab)
~~Claims~~	~~RoleAdvancedCaseManager~~	~~Advanced Case Manager~~	~~Not currently used in Pivotal. Now hidden~~
Claims	RoleCanChangeCaseManager	Can Change Case Manager	User can change the case manager assigned to a

			claim
Claims	RoleCanChangeClaimType	Can Change Claim Type	
Claims	RoleCanChangeDateClaimMade	Can Change Date Claim Made	
Claims	RoleCanChangeInjuryDateTime	Can Change Injury Date/Time	
Claims	RoleCanChangeSignificantInjuryDate	Can Change Significant Injury Date	
Claims	RoleCanDeleteNotes	Delete Claim Notes	
Claims	RoleCanEditDeleteTask	Can Edit/Delete Task	
Claims	RoleCanEditNotes	Edit Claim Notes	
Claims	RoleCanEditRestrictedClaims	Can Edit Restricted Claims	
Claims	RoleCanEnterAccClaims	Can enter claims	
Claims	RoleCanEnterAccInvoices	Can enter payments	
Claims	RoleCanEnterAccRecoveries	Can enter recoveries	
Claims	RoleCanReopenAccClaims	Can reopen Claims	
Claims	RoleCanViewRestoreNotes	View/Restore Claim Notes	
Claims	RoleCanViewTasksForAllUsers	Can View Tasks For All Users	
Claims	RoleFeeForService	Do Fee for service	NZ Only.
~~Claims~~	~~RoleIsClaimsManager~~	~~Claims Manager~~	~~Not used. Now hidden~~
Claims	RoleIsACCCaseManager	Case Manager	Was hidden, now shown. If ticked, user is case manager and shown in Case Manager list
Claims	RoleIsDataEntry	Is Data Entry	Now hidden
Claims	RoleShowInIMToDoList	Show In Tasks List	Show in Task Manager. Can have tasks assigned to them
Claims	RoleViewRestrictedClaims	View Restricted Claims	View claims that have been marked as restricted
Data	RoleCanChange	Can change claims data	Can change data that user has permission to view. Without this permission, user has read-only access to claims, members, providers etc
Data	RoleEditClients	Can Edit Clients	
Data	RoleEditContacts	Can Edit Contacts	
Data	RoleEditTeams	Can Edit Teams	

Data	RoleEditUsers	Can Edit Users	
Document Management			The Document Management section has permissions relating to document management
Document Management	RoleClientViewDocuments	View documents (for client user only)	NZ Only
Document Management	RoleDocumentAdmin	Document Admin	Can delete scanned documents
Document Management	RoleDocumentProcessing	Processing	Can begin/edit document processing
Financial	RoleApproveProviderPayments	Approve Provider Payments	
Financial	RoleApproveWBPayments	Approve Weekly Benefit payments	
Financial	RoleCanCancelPaymentBatch	Can Cancel Payment Batch	Can cancel a Westpac payment batch file
Financial	RoleCanEnterProviders	Can enter providers	
Financial	RoleEditLockedProviders	Can Edit Locked Providers	
Financial	RoleProcessPayments	Process Payments	Can create a Westpac payment batch file
Report			The Report section has permissions relating to reporting. **Note: The ability to create/edit reports, and automatic reporting is controlled by the RoleAdmin permission**
Report	RoleEditJurisdictionPhrases	Can Edit Jurisdiction Phrases	
Report	RoleEditReportGroups	Can Edit Report Groups	
Security	RoleTwoFactor	Use Two Factor authentication	Requires 2 factor authentication (e.g. SMS, email or Google Authenticator) when logging in from untrusted computer
Wiki	RoleWikiAdmin	Create and edit wikis	
~~Claims~~	~~RoleIsACCCaseManager~~	~~Case Manager~~	~~Not used~~
~~Claims~~	~~RoleSawCaseManager~~	~~SAW Manager~~	~~Not used~~
~~EOS~~	~~RoleEosQA~~	~~EOS case manager~~	~~Not used~~
~~EOS~~	~~RoleEosView~~	~~View EOS claims~~	~~Not used~~
~~Financial~~	~~RoleAuthoriseDirectCredits~~	~~Authorise direct credits~~	~~Not used~~

~~Financial~~	~~RoleCanDoChequeRun~~	~~Cheque run~~	~~Not used. Replaced by RoleProcessPayments~~
~~Financial~~	~~RoleFinancial~~	~~Financial~~	~~Not used~~
~~HealthServices~~	~~RoleHealthServices~~	~~View Health Services Claims~~	~~Not used~~
~~Motor~~	~~RoleMotor~~	~~View Motor Claims~~	~~Not used~~
~~OccHealth~~	~~RoleCanEnterOccHealthRecoveries~~	~~Can enter Occ Health recoveries~~	~~Not used. NZ Only~~
~~OccHealth~~	~~RoleOccHealthAdmin~~	~~Occ Health administrator~~	~~Not used. NZ Only~~
~~OccHealth~~	~~RoleOccHealthProducts~~	~~Can enter Occ Health products~~	~~Not used. NZ Only~~
~~OccHealth~~	~~RoleOccHealthProvider~~	~~Occ Health Provider~~	~~Not used. NZ Only~~
~~OccHealth~~	~~RoleViewOccHealth~~	~~View Occ Health claims~~	~~Not used. NZ Only~~
~~Security~~	~~RoleTwoFactorNotRequired~~	~~DON'T use Two Factor authentication~~	~~Not used. Overrides~~ RoleTwoFactor
~~Timesheets~~	~~RoleEditAllTimesheets~~	~~View/Edit all~~	~~NZ Only~~
~~Timesheets~~	~~RoleHasStaff~~	~~Has staff~~	~~NZ Only~~
~~Timesheets~~	~~RoleShowTimesheetCostCentre~~	~~Show Cost Centres~~	~~NZ Only~~
~~Timesheets~~	~~RoleTimesheet~~	~~Enter Timesheet~~	~~NZ Only~~
~~Wiki~~	~~RoleWikiEditArticle~~	~~Create and edit articles~~	~~Not used. Replaced by wiki specific permissions~~

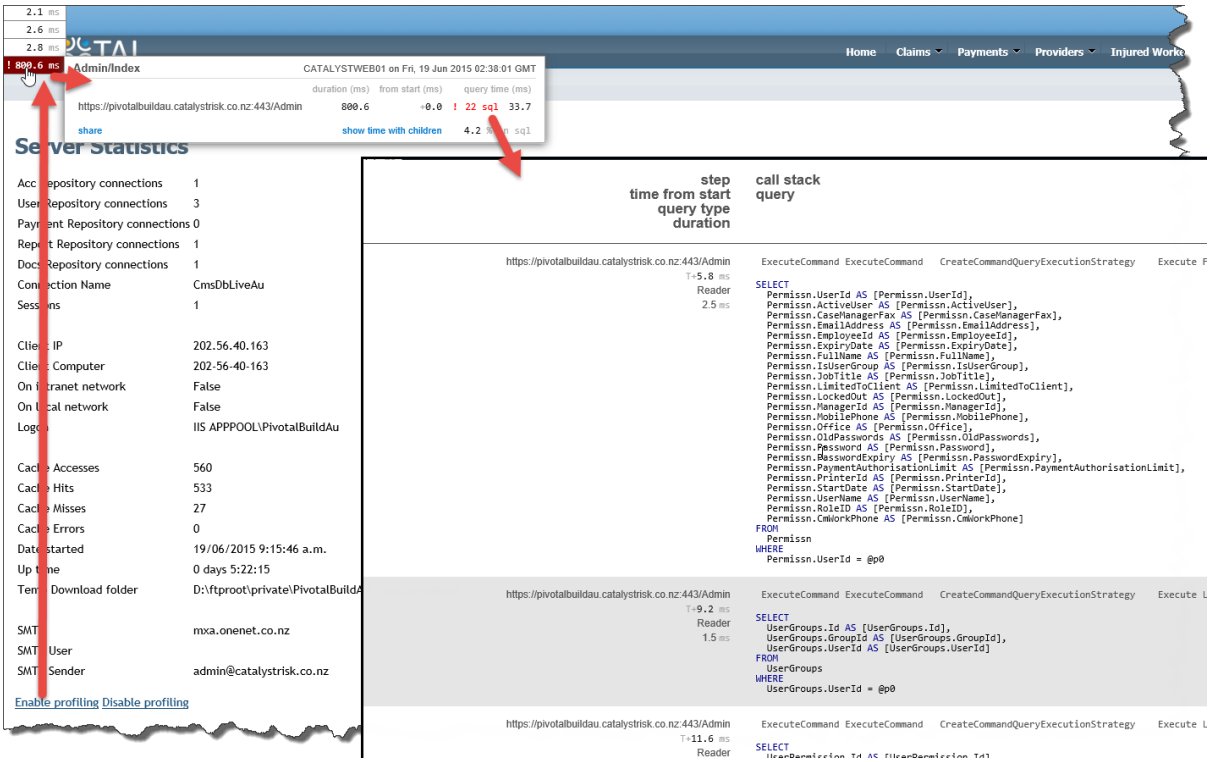
Caching

Lookup tables and infrequently changed database objects can be cached to improve performance. This is done using the CacheLayer class; defined in CacheLayer.cs and EntityCache.cs. CacheLayer supports caching and retrieving objects, and entities though the methods Add and Get.

Profiling and Performance

Profiling can be enabled by going to Admin -> Statistics and clicking on Enable Profiling:

Whenever a page is loaded, SQL profile statistics will appear in the top left. Red lines indicate duplicate queries. Click on a line to expand.



Expensive methods may be able to be sped up using Caching (either the Cache property in lightspeed, or the CacheLayer class).

E.g. InRole uses User.Permissions which hits the database. So permissions could be rewritten as:

```
public IEnumerable<Int32> Permisions
{
    get
    {
        if (_Permisions == null)
        {
            string cacheName = "UserPermissions:" + Id.ToString();
            _Permisions = Cache.CacheLayer.Get<IEnumerable<Int32>>(cacheName);

            if (_Permisions == null)
            {
                _Permisions = (from ug in UserGroups
                               where ug.Group != null
                               select ug.Group
                              ).SelectMany(g => g._UserPermissions)
                    .Union(_UserPermissions).Select(p => p.PermissionId)
                    .Distinct().OrderBy(id => id);
                Cache.CacheLayer.Add(_Permisions, cacheName, 10);
            }
        }
        return _Permisions;
    }
```

```
        }
```

Combo boxes that use ajax to load can sometimes be improved by creating the list in the view model in a PopulateLists method,

```
public virtual void PopulateLists(IAccRepository repository, User user)
```

E.g.

```
<%: Html.Kendo().DropDownListFor(m => m.RegulatorSeriouslyInjuredCode)
                .DataSource(source => source.Read(read => read.Action("GetTcds", "SharedData",
                                                    new { tcdType = TcdType.SeriouslyInjured, isFilter
Mode = false }) ))
                .UseSelectListSettings()
                .Enable(_CanEdit)
                .DropDownListWidth(400)
                .ShowHoverText()
                .HtmlAttributes(HtmlAttributes.EditorNarrowCombo)
%>
```

Can become

```
<%: Html.Kendo().DropDownListFor(m => m.RegulatorSeriouslyInjuredCode)
                .BindTo(Model.SeriouslyInjuredList)
                .UseSelectListSettings()
                .Enable(_CanEdit)
                .DropDownListWidth(400)
                .ShowHoverText()
                .HtmlAttributes(HtmlAttributes.EditorNarrowCombo)
%>
```

Where Model.SeriouslyInjuredList is set in the Action method for the page.

```
public ActionResult NewClaim4AU(int memberId)
{
    ...

        vm = new NewClaim4AuVM((Member)_Entity, m46No, injuryDate, jurisdictionCode);
    }
    vm.PopulateLists(_AccRepository, CmsUser);

    ...
}
```

In `NewClaim4AU`

```
public void PopulateLists(IAccRepository repository, User user)
{
    ...
    SeriouslyInjuredList = repository.GetSelectList(TcdType.SeriouslyInjured, selectedValue);
}
```

Translation

Some terms need to be translated between NZ and AU. To do so, use `TranslationProvider.Translate("xxx")`

Or

```
<%: Html.Translate("xxx") %>
```

Translations are stored in CmsWeb\Properties\Resources.resx. Each term is listed once for each country and sufficed with _ and the country code.

Name	Value	Comment
Menu_ACC_AU	WorkCover	https://10.8.0.25/redmine/issues/7355
Menu_ACC_NZ	ACC	https://10.8.0.25/redmine/issues/7355
ReceivedACC_AU	Received by Insurer	GeneralPV.aspx
ReceivedACC_NZ	Received by ACC	GeneralPV.aspx

Page Layouts

There are 3 main types of layout used in Pivotal. New forms and tabs should use one of these layouts wherever possible:

2 Data Columns with optional right hand action column

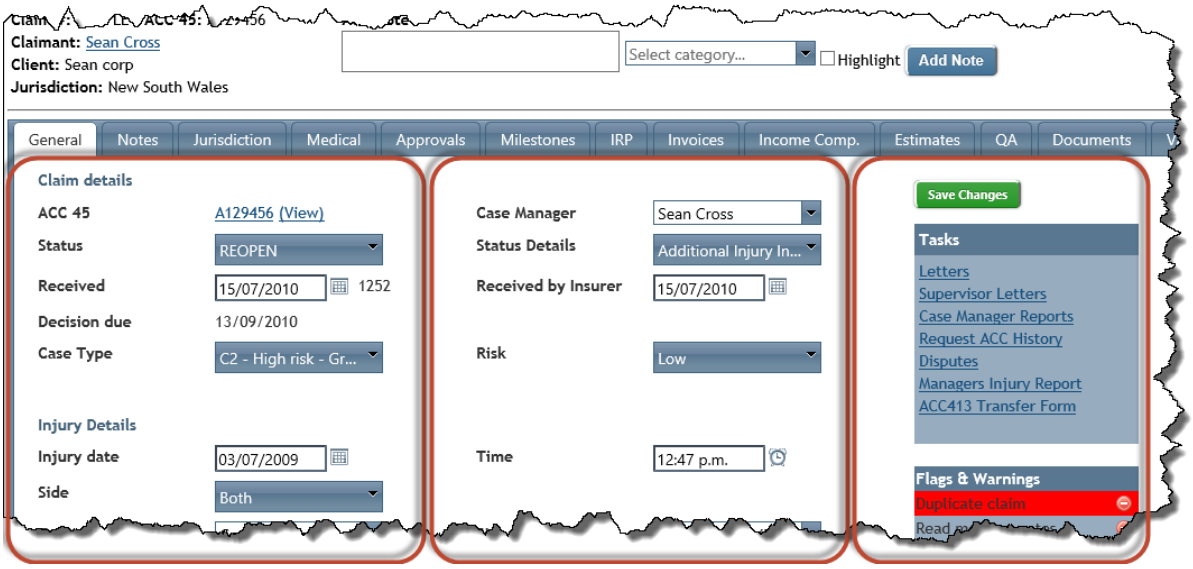
E.g. claims general page, member general page. This is used for data entry screens. The action column can display tasks and general information relating to the main item but should not include common data entry fields.

Note: data entry fields should be the same width unless this looks silly (e.g. date/time fields).

Combo boxes and dropdown lists should have a wide enough dropdown with so that the contents can be shown without word wrapping. This can be down with the JavaScript function KendoUtils.dropDownWidth.

E.g.

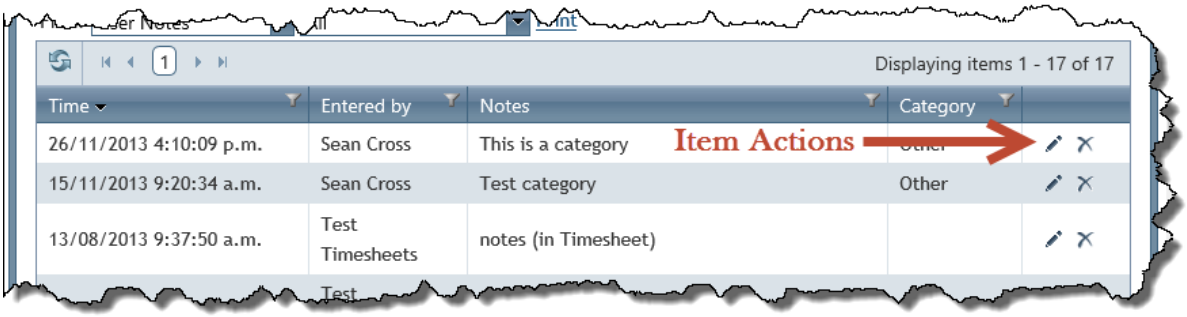
```
KendoUtils.dropDownWidth('PostalAddress_CombinedId', 500);
```



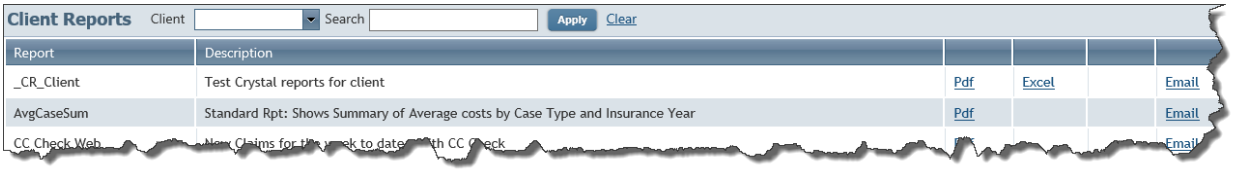
Grid with optional right hand action column

E.g. Search results, Notes, Medical, IRP, Invoices. This is used for displaying a list of related items. The Action column should have generic actions (i.e, not related to a specific item) such as Add. An action column should only be used if the grid is relatively narrow. Actions relating to a specific item in the grid should be included on the same row.

Edit actions that open a popup editor or confirmation dialog (e.g. most edit/insert/delete) should be shown as a icon, while actions that open a new page should be shown as a link.



The grid title is displayed in the left of the grid toolbar. Add New Record buttons in the right. Other toolbar items are displayed to the right of the title:



Popup layout

In most cases, grid items should be edited using a popup. This is usually a single data column form. E.g. Med Certs, invoices, notes.

Document Ready

Hovertext

Hover text (tool tips) can be enabled using the following javascript functions in

Appendix 1 – Testing notes

Developer notes

Nullable items

Do not take the value of a nullable item without checking for null, or coalescing to a value.

Bad

```
if(serviceItemId!=null&&serviceItemId!=0)
{
    si = _AccRepository.Get<ServiceItem>(serviceItemId);
    rateExGst = si.PriceExGst.Value;
    rateIncGst = GstCalculator.AddGst(si.PriceExGst.Value);
    paymentTypeId = si.PaymentTypeId;
}
```

Good

```
if(serviceItemId!=null&&serviceItemId!=0 && si.PriceExGst.HasValue)
{
    si = _AccRepository.Get<ServiceItem>(serviceItemId);
    rateExGst = si.PriceExGst.Value;
    rateIncGst = GstCalculator.AddGst(si.PriceExGst.Value);
    paymentTypeId = si.PaymentTypeId;
}
```

or

```
if(serviceItemId!=null&&serviceItemId!=0)
{
    si = _AccRepository.Get<ServiceItem>(serviceItemId);
    rateExGst = si.PriceExGst.Value ?? 0;
    rateIncGst = GstCalculator.AddGst(si.PriceExGst.Value ?? 0);
    paymentTypeId = si.PaymentTypeId;
}
```

Enums

Enums should not be mapped to/from ints. If a value uses an enum as a lookup, then it should be declared as the enum type.

Bad

```
public class InvoiceAuVM : IViewModel
 {
    ...
     public int Duplication { get; set; }
    ...
  }

    <% Html.Kendo().DropDownListFor(model => model.Duplication)
         .BindTo(Model.DuplicationList
         …
    %>


    DuplicationList = Enum
```

```
        .GetValues(typeof(Duplication))
        .Cast<int>()
        .Select(i => new SelectListItem
            {
                Value = i.ToString(),
                Text = Enum.GetName(typeof(Duplication), i),
            }
        ).ToList();
```

Good

```
public class InvoiceAuVM : IViewModel
 {
    ...
     public Duplication Duplication { get; set; }
    ...
  }

    <% Html.Kendo().DropDownListFor(model => model.Duplication)
        .BindTo(Model.Duplication.ToSelectList())
        …
    %>
```

UI Testing notes

Combo boxes and dropdown boxes

Should use dropdown list instead of combo box whenever there is a small number of items (<= 8-10)

Unless

- They all start with the same letter
- Scrolling the list is required to see all options
- Use can enter text not in the list
- Items “Should Be Space Separated” not “CamelCased”
- There is no default value that can be used
- Otherwise instructed

Should be in alphabetical order!

- Unless there is a good reason for another order

Text should not wrap (Devs: use .DropdownWidth(xxx) or .ComboWidth(xxxx))

- Unless this would require a stupidly wide dropdown

Text boxes

Test with really long text. If the text is too long for the database then an error message should be displayed alongside the text box

Display

Enum values, coded values etc should not be displayed as CamelCase, but should have spaces separating the words.

E.g. “No Capacity” instead of “NoCapacity”.

Grids

Should have a defined sort order. Normally this is by the first column

- If alpha column then sorted in ascending order
- If date column then sorted in descending order.

All columns should either be sortable or have sorting disabled for the column

All columns should either be filterable or have filtering disabled for the column

Date fields

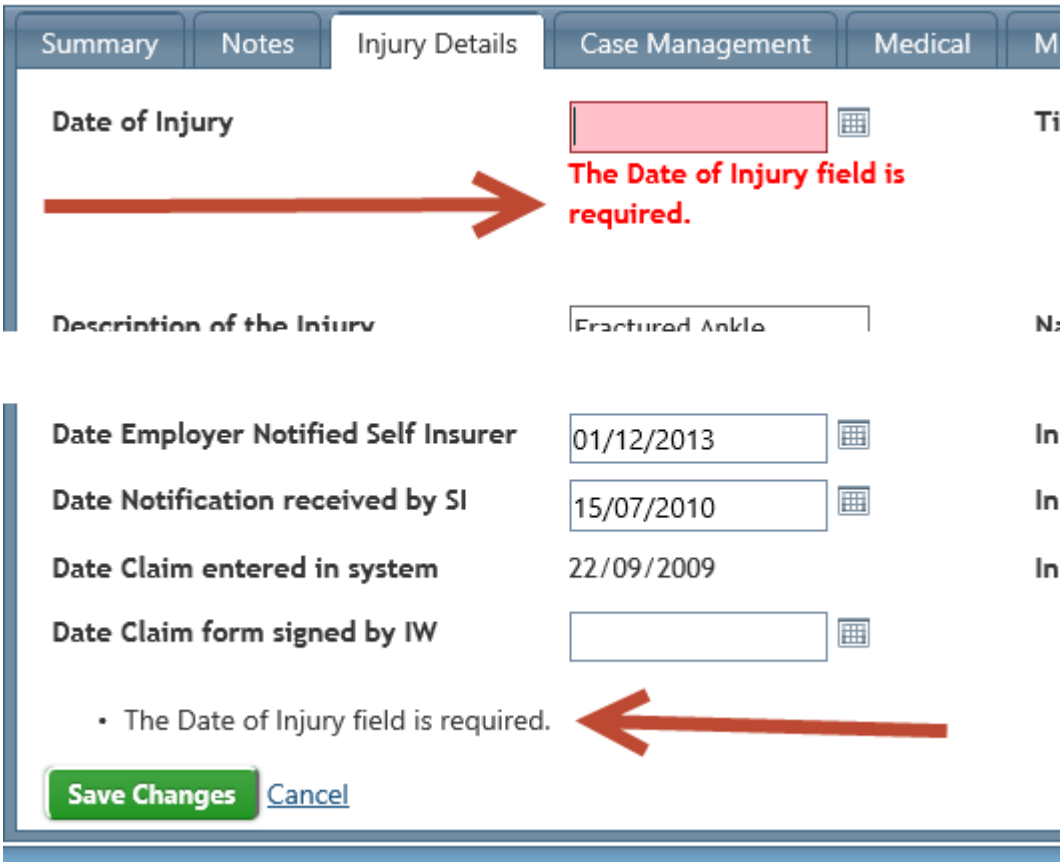
Should accept 2 digit years

Should not accept dates that are tool early, or too late. E.g. 1/1/1900 or 1/1/2100.

Validations

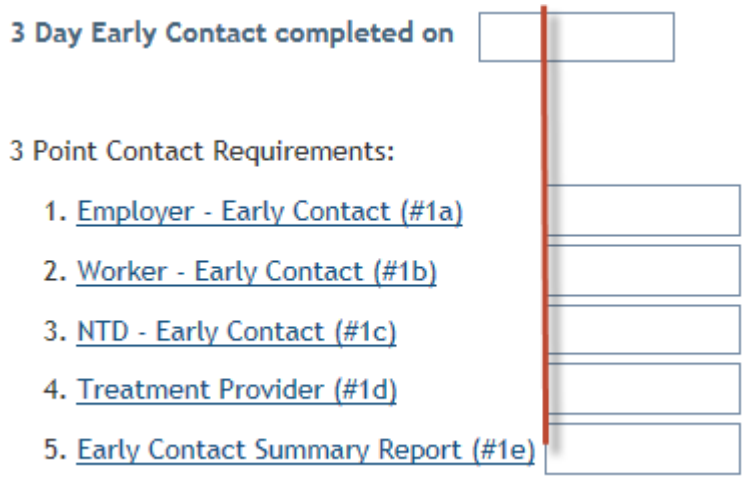
Should be red text below or beside the input editor. Not bubbles or floating text.

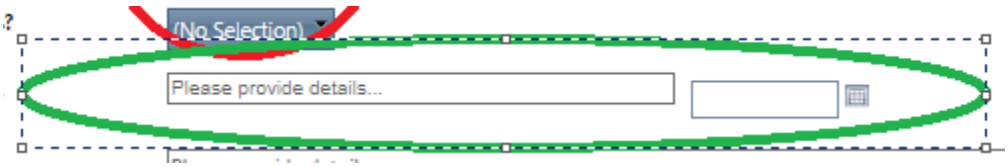
If there is space, there should be a validation summary at the bottom of the page. Not required for popups.



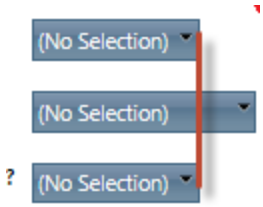
Layout

Editors and labels should be aligned both horizontally and vertically.





Editors should have a consistent width



Appendix 2 – Programming notes

Not otherwise classified.

Kendo Grids

Check boxes

Use KendoUtils.CheckBoxTemplate.

e.g.

```
columns.Bound(c =>
c.IsApproved).ClientTemplate(KendoUtils.CheckBoxTemplate("IsApproved")).Filterable(false).S
ortable(false);
```

Thumbnails

To enable thumbnails in a view:

In the link, have a class of *thumb* with attribute *rel* = thumbnail url.

```
e.g.
columns.Bound(d => d.DocumentId).ClientTemplate("<a href='" + Url.Action(
"DownloadDoc", "Document") +"/<#= DocumentId #>' class='thumb' rel='<#= ThumbnailUrl
#>'><#= ViewFile #></a>");
```

Call `Documents.showThumbs();` once the links are created

```
<script type="text/javascript">
       function dataBound(e) {
              Documents.showThumbs();
}
</script>
```

```
..      .ClientEvents(events => events.OnDataBound("dataBound"))
```

To get the thumbnail url, call `DocUtils.GetDocumentThumbnailUrl` or `DocUtils.GetDownloadThumbnailUrl`

e.g.

```
public string ThumbnailUrl
{
    get
    {
        if (DocumentId == null) { return ""; }
        var url = DocUtils.GetDocumentThumbnailUrl(DocumentId, Claim);
        return url;
    }
}
```

Data Controller

Add code to set the Selected property

```
private DocumentVM _MapDocument(Document doc)
{
    var result = Mapper.Map<Document, DocumentVM>(doc);
    result.Selected = _GetIsSelected(CmsWeb.WebConsts.Document_Reference_Selection +
doc.Reference, doc.Id);
    return result;
}

[GridAction(EnableCustomBinding = true)]
public ActionResult ReferenceDocuments(GridCommand command, string Id, string
searchTerms)
{
    var docs = from d in _DocsRepository.SearchDocuments(Id, searchTerms)
                select d;

    string ordering = string.IsNullOrEmpty(searchTerms) ? "Created" : "";
    var ga = TelerikUtils.ApplyGridCommand<Document, DocumentVM>(command, docs,
ordering, sortAscending: false, mapping: d => _MapDocument(d));
    return View(ga);
}
```

Popup windows (Kendo = new)

Use the function MvcUtils.showWindow**K** in an onclick handler)(note the trailing K)

```
<a href="#" onclick="return MvcUtils.showWindowK('/Shared/AddToDoItem?order=<%:
Model.Id %>', 'Add To-do item');">Add To-do item</a>
```

Use the following as the base form layout

```
<link href="../../Content/Site.css" rel="stylesheet" type="text/css" />

<% var guidId = Guid.NewGuid().ToString(); %>
<% using (Ajax.BeginForm(null, null, new AjaxOptions { HttpMethod = "Post", OnSuccess
= "closeWindow" }, new { id = guidId }))
    { %>


Do stuff here

<div>
    <%: TelerikUtils.SubmitButton %>
    <a href="#" onclick="closeWindow();"><span class="t-icon t-cancel"/></a> 
</div>

<% } %>

<script type="text/javascript">
    function closeWindow() {
        $('#<%: guidId %>').closest(".k-window-content").data("kendoWindow").close();
    }
</script>
```

Ajax Updating

Create a action to return the desired area

```
public ActionResult EstimateTotals(int id)
{
    AccEstimate est = _AccRepository.Get<AccEstimate>(id);
    var vm = Mapper.Map<AccEstimate, AccEstimateExtVM>(est);
    return View(vm);
}
```

In the view create a named div

```
<div id="estTotals">
</div>
```

And a function to update the div by calling the action

```
<script type="text/javascript">
       function grid_bound(e) {
              $.ajax({
                  url: '<%:Url.Action("EstimateTotals", new { Id = Model.Id })%>',
                  cache: false,
                  success: function (view) { $("div#estTotals").html(view); }
              });
       }
</script>
```

Call when required. E.g. in grid dataBound

```
<% var grid = Html.Telerik().Grid<AccEstScenarioVM>()
       …
       .ClientEvents(
              clientEvents => {
                     clientEvents.OnDataBound("grid_bound");
              })
```

Div Positioning

```
<div style="float:left;">
</div>
<div style="float:right;">
</div>
<div style="clear:both">
</div>
```

LightSpeed domain properties

The convention is that LightSpeed will look for a static field with the same name as the property with the suffix “Expression”, and of lambda expression type.

http://www.mindscapehq.com/blog/index.php/2010/09/14/ninja-domain-properties-in-lightspeed/

Here's an example:

```
// Property in Person class
public string FullName {
  get { return FirstName + " " + LastName; }
}

// Query expression in Person class corresponding to FullName property
// Note field type must be Expression<Func<entity_type,
domain_property_type>>
private static readonly Expression<Func<Person, string>> FullNameExpression
=
  p => p.FirstName + " " + p.LastName;
```

Another example

```
// Method in Person class
public int AgeInYear(int year)
{
  return year - BirthDate.Year;
}

// Query expression in Person class corresponding to AgeInYear method
// Notice second parameter of type int corresponding to "year" method
argument
private static readonly Expression<Func<Person, int, int>>
AgeInYearExpression =
  (p, year) => year - p.BirthDate.Year;
```

Sanderson style variable length list editing

See http://blog.stevensanderson.com/2010/01/28/editing-a-variable-length-list-aspnet-mvc-2-style/

In the view model, have a list property of the appropriate type. **Note the { get; set; } is required.**

```
public List<FlagExceptionVM> FlagExceptions  { get; set; }
```

Create row editor view (note BeginCollectionItem takes the name of the collection)

```
<%@ Control Language="C#" Inherits="System.Web.Mvc.ViewUserControl< FlagExceptionVM>"
%>

<tr class="editorRow t">
       <% using (Html.BeginCollectionItem("FlagExceptions"))
              { %>
                     <td>
                <%: Html.HiddenFor(m => m.Id %>
                <%: Html.DisplayFor(m => m.Flag %>
                     </td>
                     <td>
                            <%: Html.DeleteRowLink() %>
                     </td>
       <% } %>
</tr>
```

In the view model, create a table of items

```
<table id="FlagExceptionTable"  class ='tasksBlock' cellspacing="0" cellpadding="4"
border="0" >
        <tr class="tasksHeader"><td>Flags & Warnings</td><td><%: Html.AddRowLink()
%></td></tr>
        <tbody id="tableFlagExceptions">
                <% foreach (var flag in Model.FlagExceptions)
                { %>
                                <% Html.RenderPartial("FlagEditorRow", flag); %>
                <% } %>
        </tbody>
</table>
```

Add scripting for add/delete buttons

```
        <script type="text/javascript">
            $("a.addItem").live("click", function () {
                $.ajax({
                    url: '<%: Url.Action("AddFlag", "Acc") %>',
                    cache: false,
                    success: function (html) {
                      $("#tableProducts").append(html);
                      MvcUtils.stripeTable("#tableProducts");
                      MvcUtils.updateValidation("#tableProducts");
                      }
                });
                return false;
            });

            $("a.deleteRow").live("click", function () {
                $(this).parents("tr.editorRow:first").remove();
                return false;
            });
        </script>
```

In the controller, add an action for the Add function. It should return a partial view for a new row (can be the row editor view, or a new view)

`MvcUtils.updateValidation("#tableProducts");` is used to update the unobtrusive validation for the form.

A potentially dangerous Request.Form value…

This is because of something called Request Validation, that is a feature put in place to protect your application cross site scripting attacks.

You need to add the following to your action method:

```
[ValidateInput(false)]
public ActionResult MyAction (int id, string content) {
content = content.Replace("<script",
"[script").Replace("</script>","[/script]");
// ...

// or using CatalystUtils.StringUtils

content = content.StripHtml();
or content = content. SanitizeHtml();
```

```
}
```

Auto resize text areas

Use the autoresize plugin as follows

```
<script type="text/javascript">
	$(document).ready(function () {
		$("textarea#LogNotes").autoResize();
	});
</script>
```

Cascading Combos

Set CascadeFrom, Filter and ServerFiltering(true)

Use.BindTo(Model.CostCentreList) to prevent the initial loading

```
<script type="text/javascript">
	function filterInvoiceRecipients() {
		return {
			clientId: $("#ClientId").val()
		};
	}
</script

	<% Html.Kendo().ComboBoxFor(model => model.ClientId)
		.BindTo(Model.ClientList)
		.Filter(FilterType.Contains)
		.Render();
	%>
	<% Html.Kendo().ComboBoxFor(model => model.InvoiceRecipientId)
		.AutoBind(false)
		.Text(Model.InvoiceRecipientName)
		.UseSelectListSettings()
		.RequireSelection()
		.CascadeFrom("ClientId")
		.Filter(FilterType.Contains)
		.DataSource(source =>
		{
			source.Read(read =>
			{
				read.Action("InvoiceRecipients", "SharedData")
				.Data("filterInvoiceRecipients");
			})
			.ServerFiltering(true);
		})
		.Render();
	%>
```

Note for the filter function

Use

```
clientId: $("#ClientId").val()
```

to get values for a dropdownlist or an inactive combo box

use

```
clientId: $("#ClientId") .data("kendoComboBox").input.val()
```

to get the current typed text value for an active combo box

Cluetips (hover text)

Cluetip Utility functions are located in "~/Scripts/Cluetip.js" file.

1. **Cluetip.setCluetipForCombo(id)**
   Set cluetip for kendo combo, cluetip content is the full text of current selected item.
   E.g: Cluetip.setCluetipForCombo("RegulatorWorkStatus");

2. **Cluetip.setCluetipForComboWithUrl(id, url)**
   Set cluetip for kendo combo, cluetip content is the result of request using url and the current value of combo box.
   E.g: Cluetip.setCluetipForComboWithUrl("DoctorId", "/Provider/ProviderPopup");
   The method ProviderPopup should be:
   public ActionResult ProviderPopup(int selectedId)

3. **Cluetip.setCluetipForDropdown(id)**
   Set cluetip for kendo dropdown, cluetip content is the full text of current selected item.

4. **Cluetip.setCluetipForDropdownWithUrl: function (id, url)**
   Set cluetip for kendo dropdown, cluetip content is the result of request using url and the current value of dropdown.

5. **Cluetip.setCluetip: function (selector)**
   Set cluetip for elements based on selector, cluetip content will base on the rel attribute.
   E.g: Cluetip.setCluetip("a.providerLink");

There are also helper methods that can be used on a kendo combo box or dropdownlist:

```
public static ComboBoxBuilder ShowHoverText(this ComboBoxBuilder builder)
public static DropDownListBuilder ShowHoverText(this DropDownListBuilder builder)
```

Date time format and datepickers

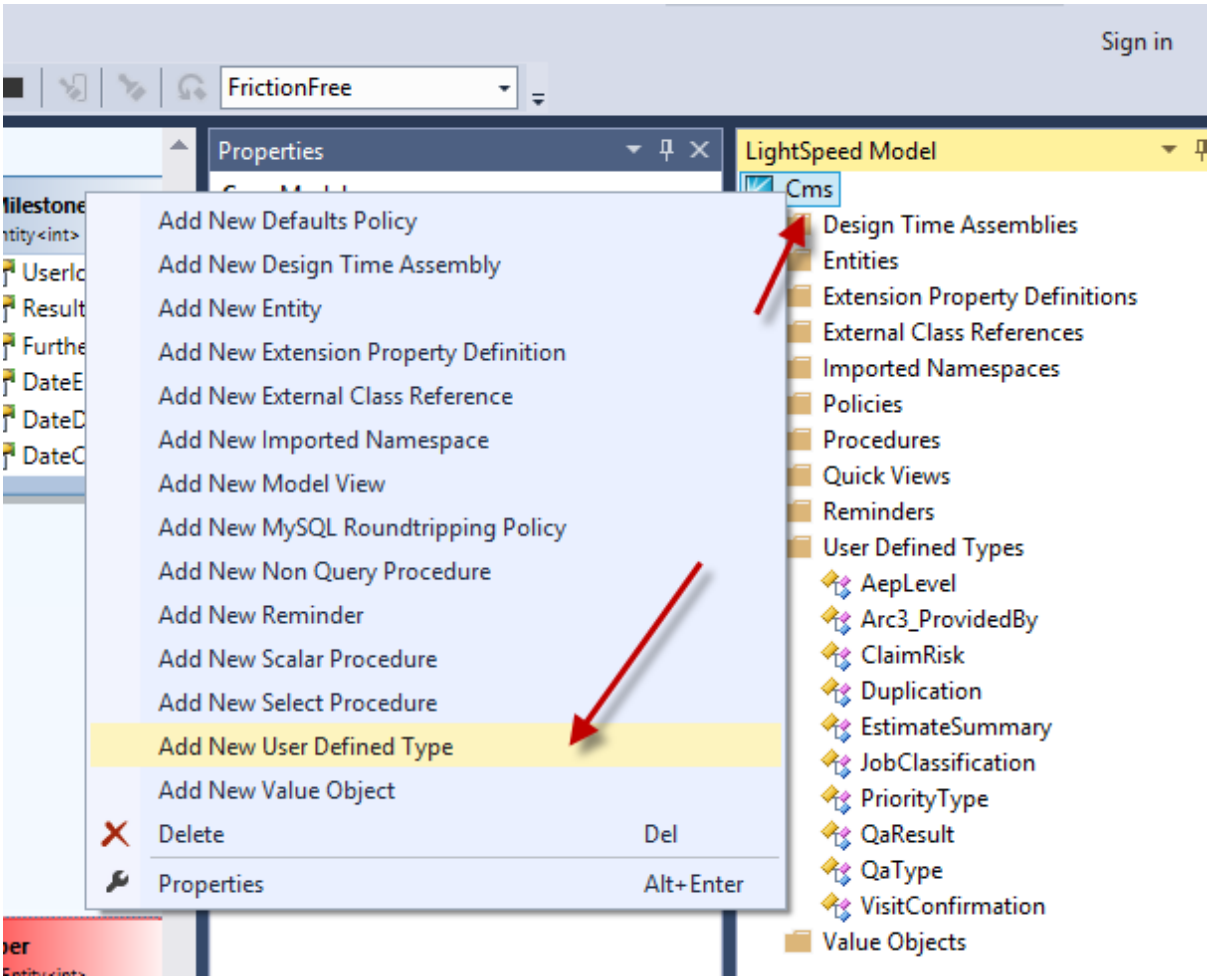
Sometimes, when a date is passed into an action and then passed back out using the same name, the datetime picker gets confused as the old format is still contained in the model state. The answer is to remove the date from the modelstate before creating the view by using ModelState.Remove().

```
[HttpGet]
public ActionResult NewClaim2(string useSelected, int? memberId, string m46No, int?
clientId, string surname, string firstnames, DateTime? dob)
{
    MemberVM vm = null;
    ModelState.Remove("dob"); // THIS
    vm = new MemberVM{ ClientId = clientId, Surname = surname, FirstNames =
firstnames, Dob = dob ?? DateTime.Today };
    return View(vm);
}
```

Defining Lightspeed properties as enums

To define a model property as an emum, do the following:

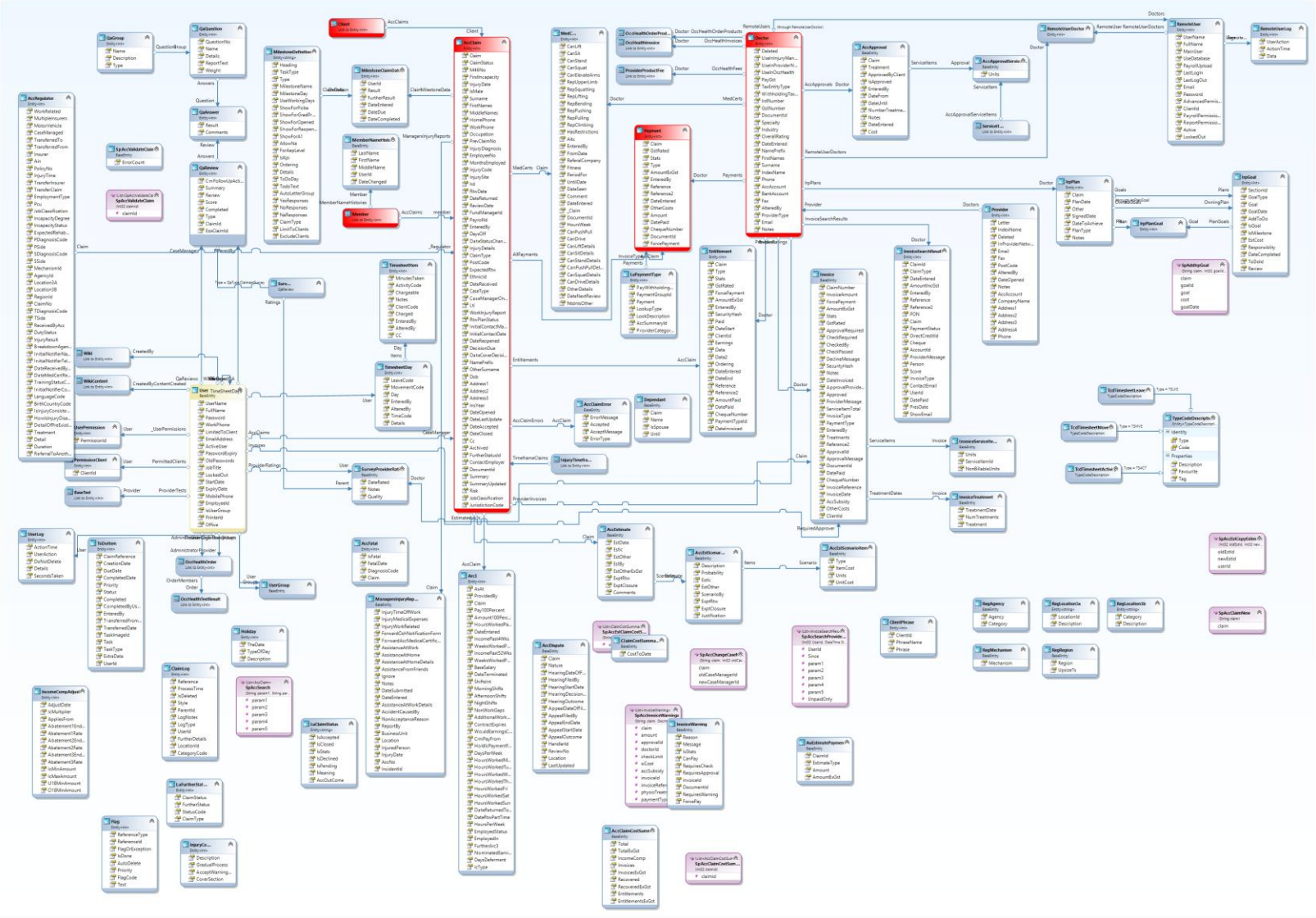
1. Open Cms.lsmodel (or the relevant lsmodel)
2. Make the Lightspeed Model window visible
3. Right click on the Cms at the top of the treeview and choose "Add New User Defined Type"



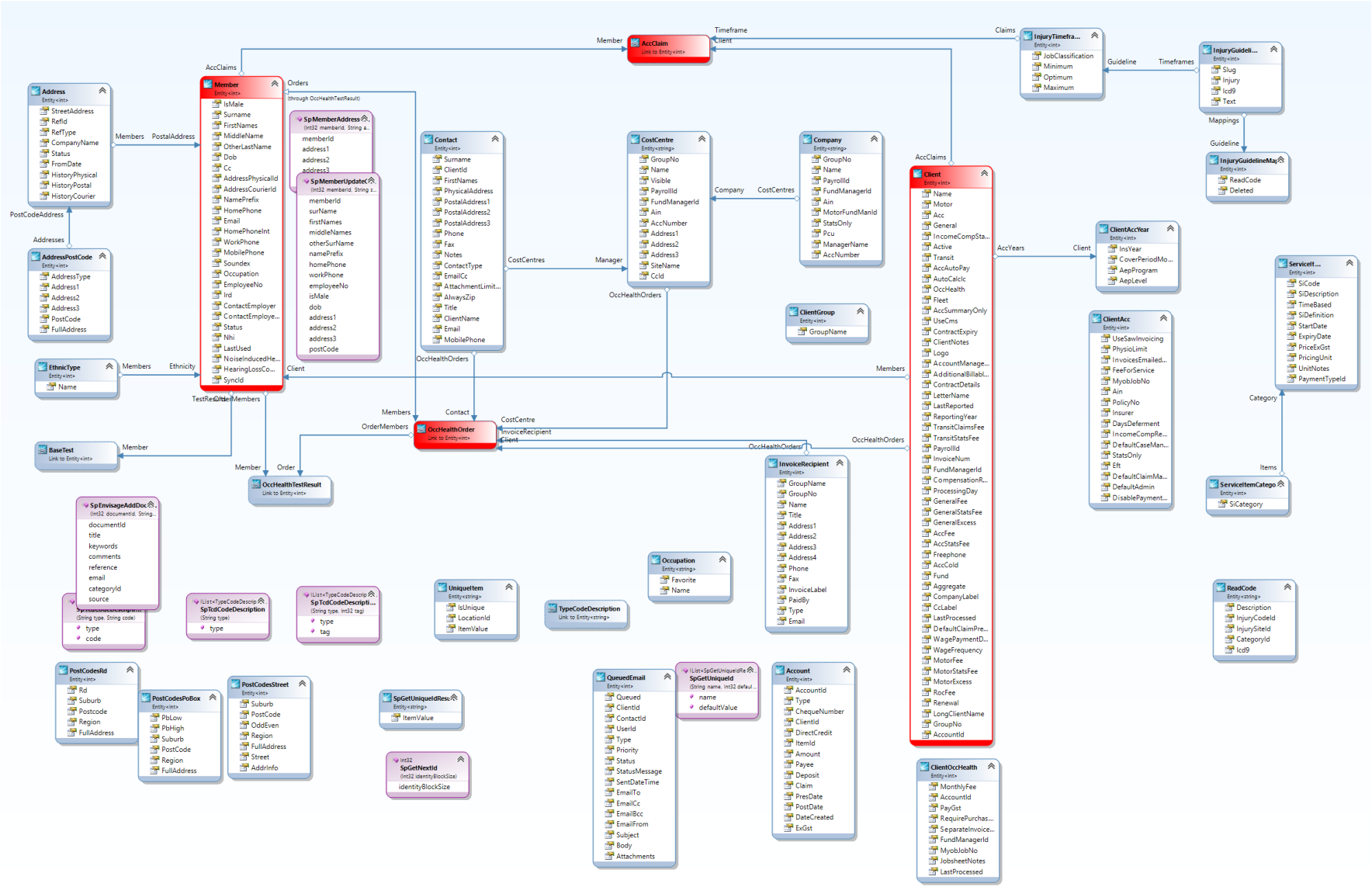
4. Set the CLR Type name to CmsModels.Entities.XXXX where XXXX is the name of the enum
5. Set the Name to XXXX where XXXX is the name of the enum
6. Find the field in the lightspeed model
7. Change the type to XXXX

Appendix 3 - ORM Model diagrams

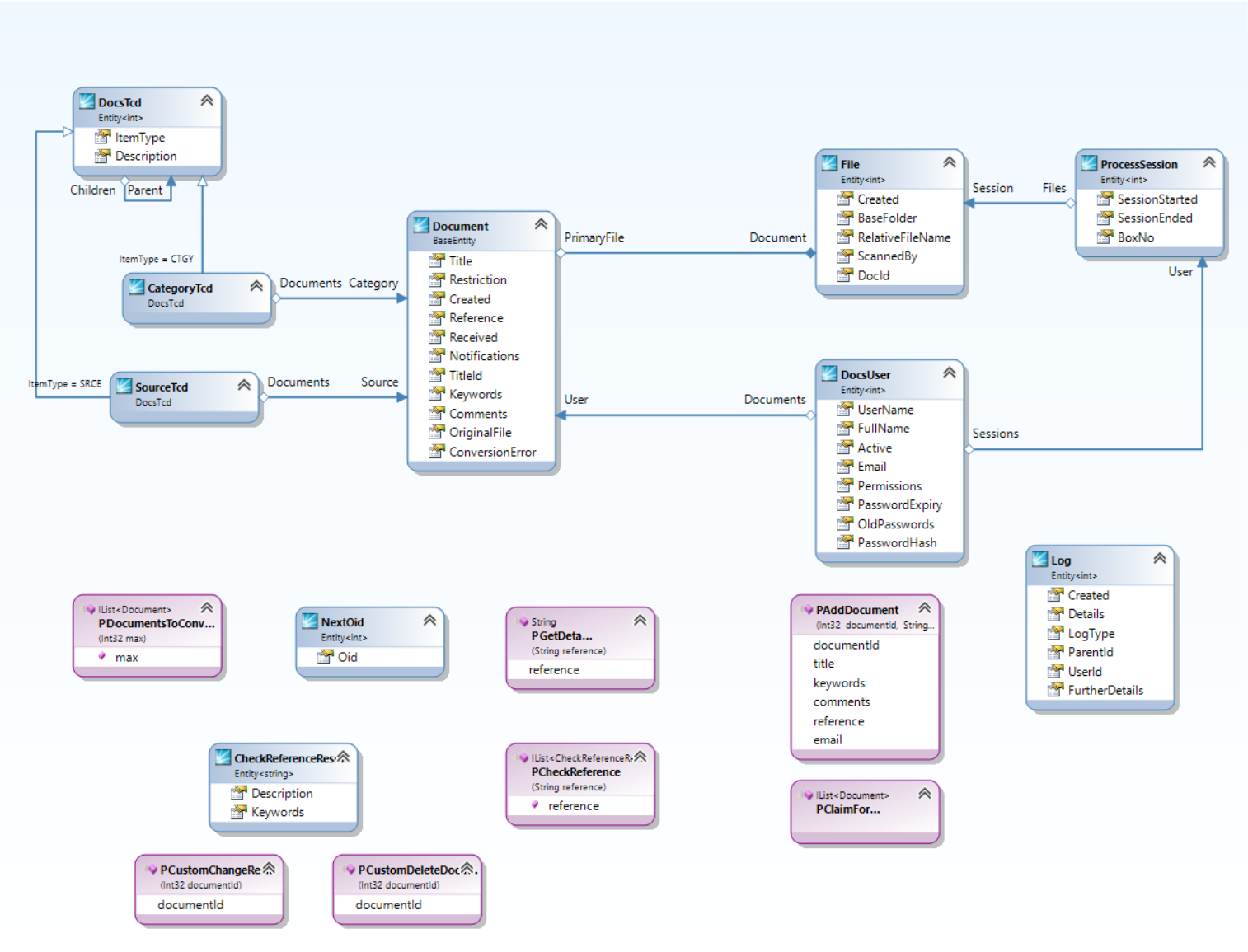
CMS Model



Shared Model



Envisage


DocsTcd
Entity<int>
ItemType
Description
Children
Parent
ItemType = CTGY
CategoryTcd
DocsTcd
Documents
Category
ItemType = SRCE
SourceTcd
DocsTcd
Documents
Source
Document
BaseEntity
Title
Restriction
Created
Reference
Received
Notifications
TitleId
Keywords
Comments
OriginalFile
ConversionError
PrimaryFile
Document
File
Entity<int>
Created
BaseFolder
RelativeFileName
ScannedBy
DocId
Session
Files
ProcessSession
Entity<int>
SessionStarted
SessionEnded
BoxNo
User
User
Documents
DocsUser
Entity<int>
UserName
FullName
Active
Email
Permissions
PasswordExpiry
OldPasswords
PasswordHash
Sessions
Log
Entity<int>
Created
Details
LogType
ParentId
UserId
FurtherDetails
IList<Document>
PDocumentsToConv...
(Int32 max)
max
NextOid
Entity<int>
Oid
String
PGetDeta...
(String reference)
reference
PAddDocument
(Int32 documentId String...
documentId
title
keywords
comments
reference
email
CheckReferenceRes
Entity<string>
Description
Keywords
IList<CheckReferenceR
PCheckReference
(String reference)
reference
IList<Document>
PClaimFor...
PCustomChangeRe
(Int32 documentId)
documentId
PCustomDeleteDoc
(Int32 documentId)
documentId

Appendix 4 – Upgrading Kendo UI

Preparatory Steps

- Back up VM
- Back up project source code
- Install specified files from \DevBin\Dependancies
    - *Examples*:
        - Telerik.ui.for.aspnetmvc.2014.1.318.commercial.msi
        - Telerik_UI_For_Silverlight_2014_1_0224_Dev.msi
- All files should be installed to C:\Program Files\ or their default directories.

Update Kendo.Mvc.dll

- Copy new Kendo.Mvc.dll from KendoUI installation folder to \DotNetDev\lib\KENDOUIMVC\<version>.
- Also include Kendo.Mvc.xml in \DotNetDev\lib\KENDOUIMVC\<version>.

Update Resource DLLs

- Copy all the resource dlls files to \DotNetDev\Projects\CmsWeb\CmsWeb\bin and overwrite files if necessary

Update Styles and Javascripts

- Create a new directory named <version> in \DotNetDev\Projects\CmsWeb\CmsWeb\Content\kendo
- Copy style files from Telerik installation folder (e.g. UI For ASP.NET MVC Q1 2014) to \DotNetDev\Projects\CmsWeb\CmsWeb\Content\kendo\<version>
- Also copy script files from Telerik installation folder (e.g. UI For ASP.NET MVC Q1 2014) to \DotNetDev\Projecs\CmsWeb\CmsWeb\Scripts\kendo\<version>
- **Where:** <version> is specified by KendoUI

Update Site.Master, Popup.Master and Error.aspx

- Find related javascript and css lines and update URL accordingly.

Reference the project dependency with new Kendo.Mvc.dll

- Remove old Kendo.Mvc dependency for CmsWeb and CmsTests and replace it with new one

(IMPORTANT) Reference scripts, styles and dlls

- Before committing, check if CmsWeb.csproj has references to new folders containing new version of Kendo UI (e.g.: 2014.1.318)

- Also, check if both CmsWeb.csproj and CmsTests.csproj has references to new xxxx.x.xxx/KendoUI.mvc.dll

Subversion

Make sure that Map files (in C:\DevDrive\DotNetDev\Projects\CmsWeb\CmsWeb\Scripts\kendo\xxxx.x.xxx\ are added to subversion.

Make sure that resource dlls are added to svn. E.g.

- Projects/CmsWeb/CmsWeb/bin/bg-BG/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/da-DK/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/de-DE/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/es-ES/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/fr-FR/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/nl-NL/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/pl-PL/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/pt-BR/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/pt-PT/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/ro-RO/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/ru-RU/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/sv-SE/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/uk-UA/Kendo.Mvc.resources.dll
- Projects/CmsWeb/CmsWeb/bin/zh-CN/Kendo.Mvc.resources.dll

Notices:

Problems:

- [8:00 AM 3/25/2014] All styles and scripts were broken in this build [#777]

Troubleshooting:

- Somehow, when the *map files are checked in, the javascripts and stylesheets work*

Remarks:

- Missing files (e.g. map and/or dll files) break the publish process resulting in an incomplete publish.
- SVN might have issues with missing files that are not properly added before they are committed. So, it's recommended that new files and folders **should be added** before they can be **committed to the server**.

Appendix 5 – SVN Guide

SVN REPOSITORIES

Documentation

http://catalystrisk.svnrepository.com/svn/Documents

Source code

http://catalystrisk.svnrepository.com/svn/Pivotal

Development

http://catalystrisk.svnrepository.com/svn/Pivotal/trunk

Release

http://catalystrisk.svnrepository.com/svn/Pivotal/branches/release

Test

http://catalystrisk.svnrepository.com/svn/Pivotal/branches/test

Custom

http://catalystrisk.svnrepository.com/svn/Pivotal/tags/...

The two repositories, which are "Documentation" and "Source Code", were created to replace the one located at ~/svn/DotNetDev and developers should pull contents separately.

ENVIRONMENTS

The purpose for this restructuring is to ensure the integrity, consistency and simplicity (yet efficiency) of the source code and controls in different environments.

Developers should work directly from the trunk and must not directly commit changes to branches/release or branches/test.

In the release branch, it should contain codes used for building application which is available to end-user and should have no major and minor defects. The codes in this branch should be merged from a number of revisions or from other branches. Thus, the typical scenario is that developers copy or merge codes from trunk to this and before this is done, they should have discussions over what to be released. At all time, the codes should be carefully managed by the team or the one accountable for it.

The testing branch should contain code used for build testable application which is primarily used by QA, QC and Business Analyst to test it functionally. Major and minor defects may be found in this environment. Normally, the code in this branch is copied or merged from a number of build-error-free stable revisions or directly from trunk. It might or might not have the latest codes from trunk.

There are also other customized environments; e.g.: unknown error testing, new feature demonstration... All should be placed in tags with the following structure tags/xxx where xxx is the unique name of a custom environment.

NAMING CONVENTION

All child folders, which are directly under tags, should be named like "yourname_your_favourite_name", which contains no whitespace or special characters except for "_".

Do not put any folders directly under branches and trunk without approval.

TUTORIALS

CLONE/CHECK OUT SOURCE CODE FOR THE FIRST TIME

1. Create a folder named "trunk" at a favourite location.
2. Right click on it and select "check out".
3. Put the source svn URL http://catalystrisk.svnrepository.com/svn/Pivotal/trunk to the textbox, leave everything default and proceed.
4. Wait for the process to complete (~150 MBs)

PREPARE ENVIRONMENT FOR THE FIRST TIME

1. Simply copy source environment to destination (by using Repo-Browser or manually)
2. Review and commit changes.

MERGE CHANGES (ONLY) WITH OTHER BRANCHES

1. Right click on the folder with your preconfigured svn environment.
2. Hover over TortoiseSVN menu item.
3. Click on "Merge…"
4. Choose either "Merge a range of revisions" or "Merge two different trees".
5. Specify options based on your current scenario.
6. Check the logs to specify the range of revision to merge
7. Check the working copy URL to make sure that the merge is correct.
8. Proceed.
9. Wait for the process to complete.

Note: Only merge changes between branches when there are existing sources. **WARNING: You might need to fix a number of merge conflicts and commit. If merge fails, revert.**

CREATE BRANCH

1. Right click on your favourite checked-out svn folder.
2. Hover over TortoiseSVN.
3. Select Branch/Tag.
4. Specify remote branch as you need.
5. Specify a copy from "Create copy in repository from".
6. Proceed
7. Wait for the process to complete.

Note: When copying data from branch to another, if the destination folder is existing, the original source folder is put inside destination folder. If non-existent, the destination contains data from source folder. **DO NOT COPY** source to destination with existing contents; use **MERGE** instead.

REMOVE BRANCH

1. Open Repo-Browser
2. Click on branch that you want to delete
3. Issue delete command

SWITCH TO ANOTHER BRANCH

1. Right click on your favourite checked-out svn folder.
2. Hover over TortoiseSVN.
3. Select Switch.
4. Choose branch or specify branch to switch to.
5. Proceed.

REVERT CHANGES AFTER MERGE

1. Right click on your favourite checked-out svn folder.
2. Hover over TortoiseSVN.
3. Select revert.
4. Tick on delete unversioned files.
5. Proceed.

Appendix 6 – Validation Standard

In this email, I want to propose a way to validate the form for kendo grid edit. We will use mainly the server side validation for this one.

- In the view, the validation messages should be created using Html.ValidationMessageFor.
- In VM class, we should mark the attribute for validation, Required, DateIsSqlSafe… For StringLength, it's optional, because it can be checked when checking the entity state is valid or not.
- In the controller action, when we update the view model to the entity, we must check ModelState.IsValid, and check if the entity is valid before saving changes. If there is any error, we return the ModelState.
- In the view, there is a handler function to show this error on the form.

In Dependants.ascx:

Add error event handler for the grid's datasource.

```
function onMemberDependantError(args) {
        KendoUtils.showGridError("DependantGrid", args);
}
.DataSource(ds =>
{
                …
            .Events(events => events.Error("onMemberDependantError"));
})
```

Add a method showGridError in KendoUtils.js, this function shows the errors for each field in the form

```
showGridError: function (gridName, args) {
      if (args.errors) {
            var grid = $("#" + gridName).data("kendoGrid");
            grid.one("dataBinding", function (e) {
                  e.preventDefault();
                  $.each(args.errors, function (propertyName) {
                  $(".k-overlay").next("div.k-widget.k-
window").find("span[data-valmsg-for='" + propertyName +
"']").removeClass("field-validation-valid").addClass("field-validation-
error").text(this.errors).show();
            });
            });
```

```
                    }
}
```

Notes:

(added if necessary)

Appendix 7 – Background Worker Implementation

1. **First off**, check if there are any repositories used by the background worker in Web.Base.Config and change their Lifestyle attribute to "Transient" to prevent mysterious failures.
2. **Secondly**, check and modify views and controllers' actions that require background worker.
3. **Thirdly**, check class IISTaskManager and use "Run" method with a lambda parameter to inject actions to background worker thread.

   Example:

```
private void StartExportProcess(HttpContext httpContext, IHubContext
hubContext, Hashtable data)

        {

            System.Web.HttpContext.Current = httpContext;

            var context = hubContext;

            bool includeLineFeeds = (bool)data["IncludeLineFeeds"];

            string reportDate = data["ReportDate"] as string;


            NswExporter exporter = new NswExporter(_NswRepository,
onExportStatus);

            ExportRecords records = new
ExportRecords(Int32.MaxValue);

            try

            {

                string filename = includeLineFeeds == false ?
"CLM297.WCA" : "NswTest.txt";

                records =
exporter.MakeRecords(DateTime.Parse(reportDate));

                records.Save(Server.MapPath("~/TempDownloads/") +
filename, includeLineFeeds);

            }

            catch (Exception ex)

            {


            }

            finally

            {
```

```
            }
```

4. **Forthly**, implement similar logic to capture and manage background worker status and events

   Example:

```
     // Prepare HttpContext, SignalR context as required
     this.exportSession = exportSession;
            var hubContext =
GlobalHost.ConnectionManager.GetHubContext<MessageHub>();
            var httpContext =
this.HttpContext.ApplicationInstance.Context;

            // There should always be Data Result (can be similar
class)
            var exportResult = new DataExportResult();

            // If a background worker requires metadata, put it as
key-value pair and add to hash table object
            Hashtable metadata = new Hashtable();
            metadata["Filename"] = includeLineFeeds == false ?
"CLM297.WCA" : "NswTest.txt";

            // Each user should be allowed to execute one task at a
time. So this method is used for check if there is a task running or
not.
            var task = IISTaskManager.GetTaskByUser(CmsUser.Id);
            if (task == null)
            {
                // If there are no tasks, create and start new
background worker
                IISTaskManager.Run(() =>
                {
                    Hashtable ht = new Hashtable();
                    ht["ReportDate"] = reportDate;
                    ht["IncludeLineFeeds"] = includeLineFeeds;
                    StartExportProcess(httpContext, hubContext, ht);
                }, CmsUser.Id, exportSession, metadata);

                // Return result
                exportResult.IsSuccessful = true;
                exportResult.FileName = metadata["Filename"] as
string;
            }
            else
            {
                if
(task.GetTaskStatus().Equals(TaskStatus.RanToCompletion) ||
```

```
task.GetTaskStatus().Equals(TaskStatus.Canceled) ||
task.GetTaskStatus().Equals(TaskStatus.Faulted))
                {
                    // Remove the task that is considered to be done
                    IISTaskManager.ListOfBgTasks.Remove(task);

                    // Unregister it on IIS Thread Pool
                    task.ForceStop();

                    // Start a new task immediately
                    IISTaskManager.Run(() =>
                    {
                        Hashtable ht = new Hashtable();
                        ht["ReportDate"] = reportDate;
                        ht["IncludeLineFeeds"] = includeLineFeeds;
                        StartExportProcess(httpContext, hubContext,
ht);
                    }, CmsUser.Id, exportSession, metadata);

                    // Return result
                    exportResult.IsSuccessful = true;
                    exportResult.FileName = metadata["Filename"] as
string;
                }
                else
                {
                    // If a background task that is initiated by a
user is running, do nothing
                    exportResult.IsSuccessful = false;
                }

            }
```

5. **Lastly**, check if the corresponding view has logic to report background worker status.

   Example of action listener and exportSession

```
private string exportSession = null;

private void onExportStatus(double d, string s)
        {
            var context =
GlobalHost.ConnectionManager.GetHubContext<MessageHub>();
            context.Clients.All.sendMessage(this.exportSession,
NswExporter.WorkerType, Convert.ToString(Convert.ToInt32(d*100)) +
"% " + s);
        }
```

   Example of base controller action

```
public ActionResult DataExport()
        {
            // Background Processing
            var task = IISTaskManager.GetTaskByUser(CmsUser.Id);

            ViewBag.JobId = task == null ? String.Empty :
task.JobId;
            ViewBag.TaskStatus = task == null ? false :
task.GetTaskStatus().Equals(TaskStatus.Canceled) ||
task.GetTaskStatus().Equals(TaskStatus.Faulted) ||
task.GetTaskStatus().Equals(TaskStatus.RanToCompletion);
            ViewBag.Filename = task == null ? String.Empty :
(String)task.Metadata["Filename"];
            //

            return View();
        }
```

Example

```
// Prepare some variables if needed
var downloadPath = "<%= currentUrl.Scheme + Uri.SchemeDelimiter +
currentUrl.Host + (currentUrl.Port != 80 ? ":" + currentUrl.Port :
"") + "/Acc/DownloadExportFile" %>";
        var downloadFile = "<%= ViewBag.Filename %>";

        // Required, this is used to connect to SignalR Hub
        var messenger = $.connection.messageHub;

        // Required, get current export session from the base action
        var currentExportSession = "<%= ViewBag.JobId %>";

        // Required, check if a process is done
        var done = "<%= ViewBag.TaskStatus %>" == "False" ? false :
true; /* Make sure that the process is done and stop listening to
other unrelated messages */

        // Prettify the view so that it won't display awkward
message
        if (currentExportSession.length == 0) {
            $("#dvLogCurrentStatus").text("Idle...");
            $("#cmdNSW").removeAttr("disabled");
        } else if (currentExportSession.length > 0 && done) {
            $("#dvLogCurrentStatus").text("Operation Completed!
Idle...");
            $("#cmdNSW").removeAttr("disabled");
            $("#dvLogDownloadLinkContainer").show();
            $("#dvLogDownloadLink").attr("href", downloadPath +
"?filename=" + downloadFile);
```

```
            $("#dvLogDownloadLink").text(downloadPath + "?filename="
+ downloadFile);
        } else if (currentExportSession.length > 0 && !done) {
            $("#dvLogCurrentStatus").text("Checking status...");
            $("#cmdNSW").attr("disabled", "disabled");
        }

        // connect to socket server
        $.connection.hub.start().done(function () {
            console.log('connection established');
        });

        messenger.client.sendMessage = function (name, type,
message) {
            /* Remove if you don't want to track background worker
log in the console */
            console.log(name + "==" + currentExportSession);
            console.log("message ==" + message)
            console.log("type ==" + type)
            /* */

            /* This section is likely to be copied/pasted */
            if (currentExportSession != null)
                if (currentExportSession == name &&
message.toLowerCase().indexOf("finished") == -1 && done == false)
                    $("#dvLogCurrentStatus").text(message);
                else if (currentExportSession == name &&
message.toLowerCase().indexOf("finished") != -1) {
                    $("#dvLogCurrentStatus").text("Operation
completed!");
                    $("#dvLogDownloadLinkContainer").show();
                    $("#dvLogDownloadLink").attr("href",
downloadPath + "?filename=" + downloadFile);
                    $("#dvLogDownloadLink").text(downloadPath +
"?filename=" + downloadFile);
                }
            /* */
        }

        $("#cmdNSW").click(function () {
            if (currentExportSession.length == 0 ||
(currentExportSession.length > 0 && done)) {
                currentExportSession = parseInt((new
Date()).getTime() / 1000).toString();
                done = false;
            }

            $("#dvLogCurrentStatus").text("Starting...");
            $("#cmdNSW").attr("disabled", "disabled");
            $("#filename").val("");
```

```
            $("#dvLogDownloadLinkContainer").hide();
            $("#dvLogDownloadLink").removeAttr("href");
            var rd = $("#datepicker").val();
            var ilf = $("#chkLine")[0].checked;
            $.ajax({
                url: "/Acc/DataExport",
                data: { reportDate: rd, includeLineFeeds: ilf,
exportSession: currentExportSession },
                type: "POST",
                dataType: "Json",
                error: function () {
                },
                success: function (data) {
                    if (data.IsSuccessful) {
                        console.log("successful");
                        downloadFile = data.FileName;
                    }
                    else
                        alert("Another task is going on");
                }
            });
        });
```

Appendix 8 – Build Process

Prerequisite:

- Basic understanding of how to configure Teamcity
- Assigned build administration role

Build Process Specification

- Based on Teamcity, found at http://202.36.68.67
- There are 7 existing builds which are categorized into database and code builds.
- The basic build steps of database can be found in each of the build details.
- The basic build steps of codes are:
    - (Clean)
    - Build
    - (Rebuild)
    - Unit Testing & Integration Testing
    - Deployment

Build Step Specification

Clean: This step should clear project build cache created by teamcity. The cache can be located at C:\Teamcity\BuildAgent\work\(Hash). This sometimes has to be done so as to avoid some asp.net build errors. You should notice that if this fails, your changes will NOT be found in the application server. This should take a 1 minute at most.

Build: This step should perform MSBuild and ASP.NET Build of project solution, which contains several dependent projects. ASP.NET Build performs building of CmsWeb.csproj. it should take up to 6-7 minutes. You should notice that if this fails, your changes will NOT be found in the application server.

Unit Testing & Integration Testing: The Build produces some testable dlls which contain test cases for both unit and integration testing. It verifies if all test cases are passed before the build can proceed. You should notice that even if this fails, your changes will be present in the application server. It should take up to 10 minutes.

Rebuild: This is currently a trivial step which is included in the build script. Mostly, it does nothing and goes to the next step. This doesn't normally fail randomly.

Deployment: This step will perform publishing of local artefacts to application server. Mostly, it ftps to remote server, compare differences and upload changed or new files as needed. If it fails, your changes will not be present in the application server. It should take up to 5 minutes.

Alternative Build Scenarios

Current:

Pivotal: Build -> Unit & Integration Test -> Rebuild -> Publish

Pivotal Sandbox: Clean -> Build -> Publish

Pivotal Daily NZ: Build -> Publish

Pivotal Daily Au: Build -> Publish

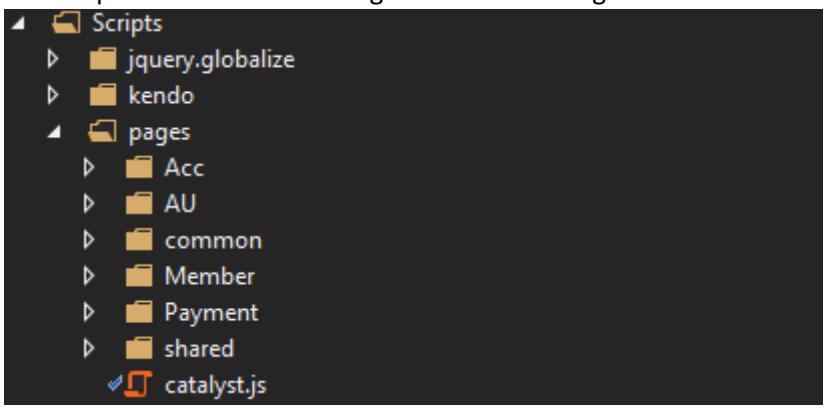
Suggested:

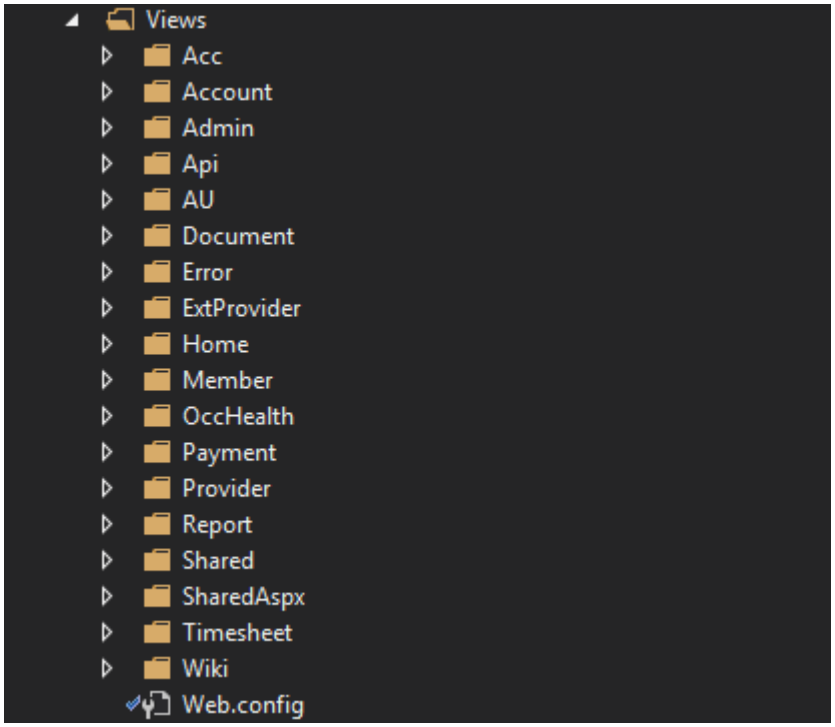
Standard: Clean -> Build -> Unit & Integration Test -> (Code Coverage) -> Publish

Appendix 9 – Javascript Coding Convention

(Originally from Toan Le)

Javascript files should be arranged in the following the structure below:



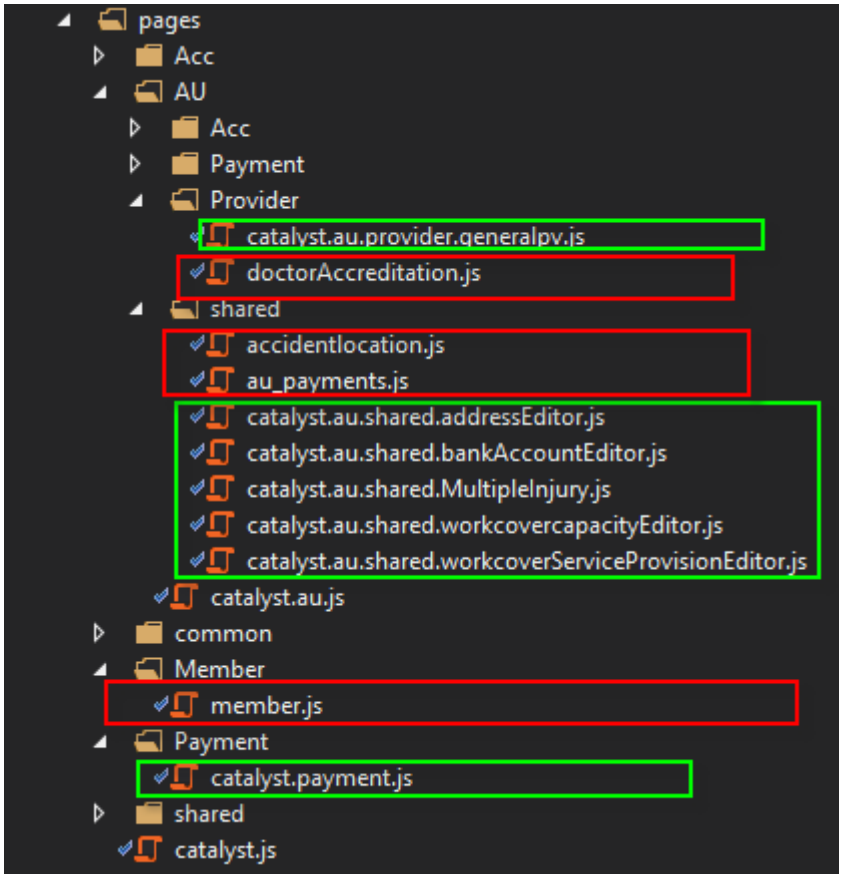


All JS files should be placed following the structure of MVC Views. That mean all JS codes from Views will be separated to another file named as naming convention:

Eg:
The User Logic code in **~/Views/AU/Shared/WorkCoverCapacityEditor.ascx** will be implemented in **~/Scripts/pages/AU/shared/catalyst.au.shared.workcovercapacityEditor.js.** The JS block is allowed only in case reference OR "**document.ready**" to Init the JS object (but it will be optimized later). **Try to keep the view clean and simplest as much as possible**.

Naming convention for js file name: The name have to reflect which view it belong to, including the namespace.

E.g: **RED** color **is invalid name**. **Green** is OK



All the JS classes implementation is required following the JS patterns such as :

```
(function ($, CataLyst, undefined) {
    // Class 'CataLyst.AU.Shared.WorkCoverServiceProvisionEditor'
    CataLyst.AU.Shared.WorkCoverServiceProvisionEditor = {
        // Region fields:
        Title: "Catalyst",
        SomeConst: 999,
        // End Region

        // Functions
        onClaimIdChange: function () {
            // do some thing
            alert("Don't kill me");
        },
        //...
    };
})(jQuery, window.CataLyst = window.CataLyst || {});
```

How to use in Kendo Binding even?

```
<td><%: Html.LabelFor(m => m.PostCodeId)%></td>
<td><%: Html.Kendo().ComboBoxFor(m => m.PostCodeId)
    .DataSource(source => source.Read(
        reader => reader.Action("GetLocalities", "SharedData", new { isPhysicalOnly = !Model.IsPostalAddress })
            .Data("CataLyst.AU.Shared.AddressEditor.FilterLocalities")).ServerFiltering(true))
    .Filter(FilterType.Contains).HighlightFirst(true).AutoBind(true)
    //.RequireSelection()   //For RM11329
    .DataTextField("FullAddress").DataValueField("Id").Filter(FilterType.Contains) // PostCodeId
    .HtmlAttributes(new { @class = "medium-editor" })%>
    <%: Html.TextBoxFor(m => m.Locality, new { @style="width:166px", @placeholder = "Enter locality..." })%>
</td>
```

One more thing, please using the DocumentReadyHelper instead of “document.ready”

```
<% string jobId = string.IsNullOrEmpty(ViewBag.JobId) ? "" : ViewBag.JobId;%>

<% Html.WhenDocumentReady("CataLyst.Payment.Process.Init(" + jobId + ")");%>
<%=Html.CreateDocumentReady()%>
</asp:Content>
```

Each JS class should have Init() Method to initialize the view.

Namespace definition